

موسسة الدراسات والبحوث العربية



الان تيسران


لغة باسكال

ترجمة د. محمد الحجار



لغة باسكال

جميع الحقوق محفوظة
الطبعة الأولى
1408 هـ - 1988 م


المؤسسة الجامعية للدراسات والنشر والتوزيع
بيروت - الحمراء - شارع النيل اده - ساحة سلام
هاتف : ٨٠٢٤٢٨ - ٨٠٢٤٠٧ - ٨٠٢٢٩٦
بيروت - المصيطة - ساحة طاهر - هاتف : ٣٠١٠٣٠ - ٣١١٣١٠
ص. ب. ٦٣١١ / ١١٣ تليفون : ٢٠٦٦٥١ E - ٢٠٦٨٠ لبنان

سلسلة بإشراف
د. عبد الحسن الحسيني

الآن تيسران

لغة باسكال

ترجمة د. محمد الحجار

المؤسسة الجامعية للدراسات والنشر والتوزيع

هذا الكتاب ترجمة :

Pascal iso / afnor

**Programmation déductive
et description de la norme**

Par

Alain TISSERANT

مقدمة

بين اللغات الموضوعة لحل أزمة مناهج السبعينات ، كان للغة الباسكال النجاح الأكبر . بعد تنظيمها ووضعها تقريباً في كل الحاسوبات ، من الميكرو الى الكبير ، يتسع إستعمالها يوماً بعد يوم مع وجود برامج تمتد من 10 أسطر الى 800 000 سطر في الستراتالات التلفونية .

إنّها أكثر سهولة من الكوبول أو الفورتران ، لكن كذلك أسرع وفعالة أكثر من أجل إبراز مفهوم ، تنقيح وصيانة البرامج .

الباسكال هي لغة للمبتدئين ، لقد تم تصوّرها بالفعل ، لتعليم البرمجة كما لو كانت مادة تعليمية دقيقة قائمة على بعض المفاهيم الأساسية الظاهرة بوضوح في اللغة .

الباسكال هي لغة للمحترفين ، إنها فعالة جداً عند التنفيذ ، وتسمح بالكتابة الجيدة لبرامج تُقرأ بسهولة ، فإذن قابلة للتعديل من شخص آخر وسهلة التنقيح .

لدى هذا العمل ، الطموح بأن يكون قد استجاب لطلبات المتهنين وكذلك قد استفيد منه كدعامة للمادة التعليمية . لكي يكون فعالاً إعتدنا العرض التدريجي لذا نراه ، يحتوي على تكريرات مفتعلة ، لكن يستخدم أسلوباً وتنويطات محددة . نجد فيه الكثير من الأمثلة ، لكن القليل من التفصيلات التكنولوجية .

تتطابق اللغة المشروحة مع النظم ISO وAFNOR

يسمح الجزء الأول للمبتدئ ، بأن ينطلق في عالم البرمجة مستعملاً طريقة أثبتت وجودها . نشرح في الجزء الثاني الأدوات المعالجة ومن ثم في الجزء الثالث طريقة مُعالجتها ؛ إن العناصر المدخلة في هذين الجزئين تتماشى مع كل ما هو ضروري للبرمجة في جميع الأوقات . يُتمم الجزء الرابع المفاهيم التي سبقته بهدف إستعمال لغة الباسكال بشكلها الكامل . أخيراً ، يوجز الجزء الخامس اللغة الى مُذكّرة مساعدة . تقوم بعض الملحقات بإضافة بعض المُتمّمات ، وتسهّل البلوغ المباشر في الكتاب .

لقد عرّفت لغة الباسكال ، المحدّدة عام 1969 من قبل نيكلوس ويرث Niklaus Wirth كما لو كانت تطبيقاً للبرمجة المركّبة ، إنتشاراً سريعاً . اليوم يوجد نظام عالمي يحدّد اللغة ؛ مع ذلك فإنّ معالجي الباسكال الموجودين لا يخضعوا جميعهم لهذا النظام ؛ فإذن أصبح من الواجب على القارئ ، إدخال تعديلات طفيفة على البرامج المعطّية كأمثلة ، قبل إستعمالها على حاسوبه المفضل .

الفصل الأول

كيفية البدء بكتابة البرنامج

0.1 - التحليل

يراد من هذا الفصل أن يكون مدخلاً للمبتدئين في البرمجة : قبل الدخول في تفاصيل كتابة البرامج في لغة الباسكال «Pascal» ، أو في أية لغة أخرى ، يجب الإلمام ببعض مبادئ التحليل .

تعني كلمة التحليل في عرضنا هذا فن العبور بمشكلة الى برنامج يحلها . إن ذلك يستوجب أنشطة عقلية ليست دائماً سهلة لكنها ضرورية ، والتي يمكن جعلها سلسة أكثر بوضع أنفسنا ضمن إطار محدد : الطريقة الإستنتاجية . إنها طريقة شديدة السهولة بحيث تتيح تعليم سريع ، كما أنها فعّالة بحيث تسمح بإدخال المفاهيم العامة والمفردات والمصطلحات الأساسية للبرمجة .

سيتم تناول كتابة البرامج في لغة الباسكال بالمعنى الحرفي للكلمة ، في الفصل الثاني . هذا لا يمنع الحصول منذ الآن على برامج « تعمل » .

1.1 - الإنطلاق من النتيجة

لنفرض المسألة التالية : « أَّجَرِ فَوْتَرَة على ميزان مسَّجَل » . بطرحها بهذه الطريقة ، فإن المسألة تجرنا إلى عدة أسئلة : ما هي وحدات القياس المستعملة (كيلوغرام ، طن ؟) ، كيف يتم حساب الثمن المتوجب دفعه عند إتمام الوزن (هل هناك من مبلغ إضافي ثابت ، ضريبة ، حسم ؟) ، كم يوجد من وزنات متتالية ، إلخ . . . إن الإجابة على هذه الأسئلة تعني العبور من المشكلة (المسألة الأساسية) إلى المواصفة أي حل مُضمَرات المسألة .

للإجابة على الأسئلة ، يجب العمل على استدعاء مجموعة من المعلومات الخاصة بالمسألة المطروحة ؛ يمكن أن ينتج عن ذلك أنه من الضروري مراجعة كتاب ما ، أو مراجعة متخصص في ذاك المضمار (الرياضيات ، المحاسبة ، الميكانيك ، . . .) . إذا كانت المواصفة غير كاملة أو مغلوبة ، فإن البرنامج لن يؤدي بالتالي الى نتائج جيدة .

يمكن أن تكون المواصفة في مثلنا هذا هي التالية : « إحص الثمن المتوجب دفعه (عدد صحيح من القروش) بعد إجراء وزنة ؛ الثمن هو حاصل ضرب الوزن (عدد صحيح من الغرامات) بالسعر الإفرادي (عدد صحيح من القروش للكيلوغرام الواحد) » .

إن المواصفة تحدد نصوص المسألة والحسابات الواجب إجراؤها لحلها ، لكنها غير معنية بتفاصيل الحسابات ، ولا بترتيب العمليات : إن هذا هو هدف كتابة التحليل الذي يُترجم فيما بعد وبطريقة سهلة جداً وأتوماتية إلى برنامج .

مسألة ← مواصفة ← تحليل ← برنامج

التحليل

يتألف التحليل من تعريفات ، معادلات تُعَيَّن قيمة إنطلاقاً من ثوابت وقيم أخرى (الثمن = الوزن * السعر الإفرادي) ، ومن معجم ، فهرس لأسماء (= معرفين) معطية للقيم المتوجب حسابها : 1000

المعجم	التعريفات
3 - الثمن (صحيح) : الثمن المتوجب دفعه من قبل الزبون ، بالقروش	نتيجة = « اكتب » ثمن
2 - الوزن (صحيح) : الوزن بالغرام	الثمن = الوزن * $\frac{\text{السعر الإفرادي}}{1000}$
1 - السعر الإفرادي (صحيح) : السعر بالقروش للكيلوغرام	الوزن ، السعر الإفرادي وفقاً لمعطية
	ترتيب التعريفات

الطريقة الإستنتاجية (Méthode déductive)

لتكوين تحليل ، تقترح الطريقة الإستنتاجية الإنطلاق من تعريف النتيجة للصعود نحو المعطيات .

القاعدة 1 :	نُحدّد « النتيجة » ؛ إن تعريفها هو العنصر الأول الذي يجب إدخاله في التحليل ؛
القاعدة 2 :	ومن ثم ، « في كل مرحلة » ، نستبدل أحد المعرفين الذين لم يحددوا بعد من المعجم بـ : - في قسم التعريفات ، تعريف المعرف ، وذلك ، إذا كان ضرورياً ، بإدخال معرفين جدد - في قسم المعجم ، المعرفين الجدد ، وذلك حتى يصبح كل معرفي المعجم قد تم تحديدهم .
القاعدة 3 :	في النهاية نحدد ترتيب التعريفات (من المعطيات إلى النتيجة) .

(نستنتج من كل تعريف مجموعة أسئلة التي بدورها تؤدي الى تعريفات جديدة) .
هكذا فإن الترتيب الأولي المتبع في كتابة التعريفات لا يهّم أمر الترتيب النهائي المتبع في عملية الحساب .

يمكن أن يكون التعريف خاصاً بقيمة داخلية ، مثلاً الثمن = الوزن *
السعر الإفرادي ، أو خاصاً بقيمة خارجية تكون إما نتيجة (نتيجة = « اكتب » ثمن) ، إما معطية (الوزن ، السعر الإفرادي « وفقاً » لمعطية) .

- فيما يخص المعجم ، فإنه يتم وصف كل معرف بـ :
- نوع القيمة الذي يمثل (صحيح ، حقيقي ، سمة ، ...)
- وبملاحظة موجزة على أن يتم تحديد إستعمالها .

- فيما يخص التعبير ، القسم الأيسر من التعريف ، نجد :
- مجموعة معرفين ، مثل سعر ، وزن
- مجموعة ثوابت ، إما صحيحة مثل 100 ، 7 -
إما حقيقية مثل 3.14159 ، 0.05 -
- مجموعة مؤثرات ، مثل + ، - ، * (عملية الضرب) div (قسمة صحيحة : النتيجة تكون قيمة صحيحة) .

للمعرف تعريف وحيد ، لذا يتم في المعجم ، شطب المعرفين الذين سبق تعريفهم .

مثال

المرحلة 1 القاعدة 1 (تعريف « النتيجة »)

التعريفات	المعجم
نتيجة = « أكتب » ثمن	الثمن (صحيح) : الثمن المتوقع دفعه من قبل الزبون ، بالقروش
(« نتيجة » إنه معرّف سبق تحديده)	

المرحلة 2 : القاعدة 2 (تعريف الثمن)

التعريفات	المعجم
نتيجة = « أكتب » ثمن	الثمن (صحيح) : الثمن المتوقع دفعه من قبل الزبون ، بالقروش .
الثمن = * السعر الإفرادي 1000 div	الوزن (صحيح) : الوزن بالغرام
	السعر الإفرادي (صحيح) : السعر بالقروش للكيلوغرام

المرحلة 3 : القاعدة 2 (تعريف الوزن والسعر الإفرادي)

التعريفات	المعجم
نتيجة = « أكتب » : ثمن	الثمن (صحيح) : الثمن المتوقع دفعه من قبل الزبون ، بالقروش
الثمن = الوزن * السعر الإفرادي 1000 div	الوزن (صحيح) : الوزن بالغرام
الوزن ، السعر الإفرادي « وفقاً » لمعطية	السعر الإفرادي (صحيح) : السعر بالقروش للكيلوغرام

المرحلة 4 : القاعدة 3

كل معرّف في المعجم قد تم تعريفهم ، بذلك نعمل على ترتيب التعريفات :

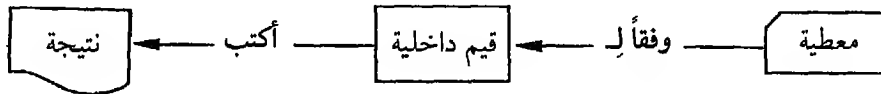
« قاعدة : لا يمكن إستعمال معرف (في القسم الأيمن) قبل أن يتم تعريفه (في القسم الأيسر) » .

الإنطلاق من النتيجة

إن فكرة الإنطلاق من تعريف النتيجة الواجب الحصول عليها (إنها فكرة مشتركة مع كثير من طرق التحليل فمثلاً طريقة وارنيه Warnier تنطلق من وصف شكل السجل الحاوي على النتيجة) ليست إلا تطبيق للطريقة الديكارتية المعتمدة على تقسيم كل مسألة إلى عدة مسائل ثانوية مستقلة عن بعضها البعض والتي بدورها نجزّها حتى الحصول على مسائل سهلة المعالجة .

ما نفعله هنا هو تعريف معرف ما بواسطة معرفين آخرين والتي تكون تعريفاتهم مستقلة عن بعضها : نعطي اسماً لكل مسألة ثانوية للإهتمام بها وحدها فيما بعد . لن نهتم بأمر ترتيب تنفيذ الحسابات قياس الوصول الى مرحلة التنظيم .

ملاحظة : إن « النتيجة » و « المعطية » هما معرفان محددان مسبقاً ويسمحان بالإتصال مع خارج البرنامج فمثلاً « النتيجة » يمكن أن تكون الشاشة (أو اللوحة) ، والمعطية يمكن أن تكون الملامس Clavier (أو البطاقات) :



2.1 - البرمجة في لغة الباسكال

إن التحليل الذي أجريناه يتطابق مباشرة مع برنامج باسكال

1 : يُكتب عنوان برنامج :

برنامج معرف (دُخْل ، خَرَج) ؛ **Program identificateur** (input , onput) { ملاحظة تحدد هدف البرنامج } .

حيث أن المعرف المختار هو غير ذي معنى بالنسبة لباقي البرنامج .

2 : يترجم المعجم بمجموعة من التصريحات تبدأ بالكلمة - المفتاح **Var**

Var معرف : النوع ؛ { ملاحظة } **Var identificateur: type; / Commentaire**

.....

معرف : النوع ؛ { ملاحظة }

.....

تُترجم الأنواع صحيح ، حقيقي ، سمة على التوالي بـ **char, real, integer** (كما نلاحظ فإن سَحوْلغة الباسكال هو إنكليزي) .

3 : تُترجم التعريفات الواحدة تلو الأخرى حَسَبَ الترتيب المُتَّبِع بواسطة عبارات ؛ يبدأ هذا القسم بكلمة «begin» وينتهي بكلمة «end» .

- تعريف سهل مثل $a = b + c$ يُكتب على الشكل التالي $a = b + c$:
 - تعريف لنتيجة مثل : نتيجة = اكتب x يُكتب : `writeln (x)`
 - تعريف لمعطيات مثل `v, u, t` « وفقاً » لمعطية يكتب `read (t, u, v)`
 - عبارتين متتاليتين يتم فصلهما بواسطة نقطة - فاصلة .

إن كتابة برنامج باسكال مسألة سلسلة كفاية بحيث : يمكن إضافة تباعدات (إلا في المعرف أو في رمز اللغة) ، يمكن إزالتها (إلا في حالات الإلتباس ، مثلاً بين begin و ... = a : يجب وجود تباعد واحد على الأقل) ، إضافة أو إزالة قفزات على السطر . ما يجب تأمينه هو قراءة ممتازة للبرنامج .

مثال : الميزان

en-tête	<code>program balance(input,output);</code>	
déclarations	<code>{ فوترة على ميزان مسجل var prix:integer;</code>	{ الثمن المترجم دفعه من الزبون
	<code>poids:integer;</code>	{ بالقروش { الوزن بالغرام
	<code>prixUnitaire:integer;</code>	{ الثمن بالقروش { للكيلوغرام
énoncés	<code>begin read(poids,prixUnitaire); prix:=poids*prixUnitaire div 1000; writeln(prix) end.</code>	

(en-tête : عنوان ؛ Prix : الثمن ، Prix unitaire : سعر إفرادي ، déclarations : تصريحات ، Poids : الوزن ، énoncés : عبارات ، balance : ميزان)

التنفيذ

يتطلب هذا البرنامج معطيتين صحيحتين ويكتب نتيجة صحيحة : إذا أعطيناها 500 ، 2000 فإنه يطبع النتيجة 1000 .

يجري إدخال المعطيات تبعاً للترتيب المطلوب في البرنامج (في حالتنا هذه فإن الوزن يساوي 500 غرام والسعر الإفرادي يساوي 2000 قرش / كلغ . هاتين المعطيتين هما مفصولتين عن بعضهما بواسطة عدد من التباعدات أو برجوع إلى السطر ، دون تحديد للوحدة . كذلك فإن النتيجة تُكتب على شكل قيمة (1000 قرش) (1) .

(1) في بعض الحاسبات الآلية ، تكون السمة الأولى لكل سطر مطلوب طباعته ، سمة تحكّم لتقديم الورق ؛ وبذلك فإننا سنكتب `writeln (' ', Prix)`

ملاحظة : تبعاً لنوعية الحاسب الآلي المستعمل ، فإن التمثيل الشكلي للأحرف في البرنامج يمكن أن يتغير دون تشويه للمعنى :

Prix يمكن أن تكتب Prix, PRIX, Prix, prix, prix إلخ . .
 Pesé يمكن أن تكتب PESE, pese, pesé ، . . .
 إلخ . . (أنظر 2.1) .

التعبير (Expressions)

يتألف التعبير في لغة الباسكال من :

- متغيرات ممثلة بواسطة معرفين ؛ بكل معرف يتم ربط نوع (integer مثلاً) وفي كل لحظة ، قيمة مأخوذة في النوع .
- مع $\text{Var } i : \text{integer}$ ؛ فإن العبارة $i := i + 1$ تعني « خذ قيمة المتغير i (صحيحة) ، أضف إليها القيمة (الصحيحة) 1 ، هذا ما يعطي القيمة الجديدة (الصحيحة) للمتغير i » ؛
- ثوابت ؛

● مؤثرات (Opérateurs) :

- على متأثرين (Opérande) من النوع integer (صحيح)، تعطي المؤثرات + ، - ، * (الضرب) ، div (قسمة صحيحة) ، mod (حاصل قسمة صحيحة) جميعها نتيجة من النوع integer .
- على متأثرين من النوع real (حقيقي) : + ، - ، * ، / (قسمة حقيقية) يعطوا نتيجة من النوع real .
- على متأثر من كل نوع integer أو real تعطي المؤثرات + ، - ، * ، / نتيجة من النوع real (جرى تحويل المتأثر integer إلى النوع real قبل العملية الحسابية) .
- / هي دائماً عملية حسابية على أعداد حقيقية .

فمثلاً 3 div 7 تساوي 2 و 3 mod 7 تساوي 1 (صحيح)

3 / 7.0 ، 7/3.0 ، 7/3 ، 3.0 / 7.0 يساوي 2.5 (حقيقي)

إن تقسيم التعبير يتم من الشمال إلى اليمين :

2 — 3 — 7 تعني 2 — (3 — 7)

لكن المؤثرات الضاربة (* ، / ، div ، mod) لها الأسبقية على المؤثرات

الجامعة (+ ، -) :

2 * 4 + 3 * 7 تعني (2 * 4) + (3 * 7)

يمكن دائماً تحديد ترتيب التقييم للتعبير وذلك بالوضع بين قوسين :
 $2.5 / 3.0 / 7.5$ تعني $2.5 / (3.0 / 7.5)$ أي $(3.0 * 2.5) / 7.5$.

التعيين (Affectation)

لا يمكن التعيين لمتغير صحيح إلا قيمة صحيحة :
 $a = a / 2$ هي غير قانونية ($a / 2$ هي حقيقية)
 لا يمكن التعيين لمتغير حقيقي إلا قيمة حقيقية أو قيمة صحيحة (فإذن هناك تحويل أوتوماتي سوف يتم) :

$7 \text{ div } 3 \approx X$ تعين للمتغير الحقيقي X القيمة 2.0
 سوف يتم تفصيل هذه القواعد في الفصل الثاني .

3.1 - التكرارية مع عداد (Iteration avec Compteur)

يتم الحساب في كثير من الحالات ، على متسلسلات من القيم وليس فقط على قيم بسيطة .

من الناحية العامة ، نستعمل المتسلسلة (Suite) إما لمعرفة كل حدودها ، مثلاً اكتب متسلسلة من الفواتير ، إما لمعرفة الحد الأخير ، مثلاً اكتب المبلغ المتوجب دفعه بعد عدة وزنات . يتم الحصول على عناصر المتسلسلة بواسطة التكرارية ؛ يحسب الحد ذي الدليل i من متسلسلة :

- إما بشكل مستقل عن الحدود الأخرى $u_i = f(k)$
 (مثال : اكتب متسلسلة من الفواتير ، الناتجة عن وزنات مستقلة)
- إما تبعاً للدليل الخاص به : $u_i = f(i)$
 (مثال : اكتب جدولاً للضرب بـ $i : N$ بـ $t_i = N * i$)
- إما تبعاً للحد السابق : $u_i = f(u_{i-1})$
 (مثال : المبلغ الإجمالي المتوجب دفعه : $S_i = S_{i-1} + \text{Prix}_i$)

أخيراً يمكن أن يكون عدد الحدود معروفاً صراحة (اكتب 20 فاتورة) ، أو محدّد بخاصة (نوقف جمع حدود المتسلسلة عندما يصبح الباقي أقل من قيمة معطيه) . تؤدي الحالة الأولى الى تكرارية مع عداد :

$$u = \text{dernier } u_i = f(k, i, u_{i-1}) \text{ pour } i \text{ de } d \text{ à } a$$

($\text{dernier} = \text{آخر}$ ، $\text{pour } i \text{ de } d \text{ à } a$ = « الحلقة لـ i »)
 بينما الحالة الثانية تؤدي إلى تكرارية مشفوعة بشرط توقف :

$$u = \text{dernier } u_i = f(k, i, u_{i-1}) \text{ pour } i \text{ de } d \text{ inclus arrêt}$$

(arrêt = توقف ، inclus = مضمَّن)
إذا كان الحد الأخير مضمَّن في الحساب

$u = \text{dernier } u_i = f(k, i, u_{i-1})$ pour i de d tant que non arrêt

(tant que = طالما « الحلقة طالما »)
إذا كان الحد الأخير مستثنى في الحساب

التكرارية للكتابة

تؤدي المسألة « إجراء n فوترة على ميزان مسجل » الى المواصفة « إحسب الثمن المتوقع دفعه (عدد صحيح من القروش) بعد كل من الـ n وزنات ؛ الثمن هو . . . (أنظر 1.1) المطلوب إذن هو كتابة نتيجة سهلة لعدد n من المرات

résultat = écrire prix pour pesée de 1 à n

(résultat = نتيجة ، écrire = اكتب ، prix = ثمن ، pesée = وزنة)

رُقمت الوزنات من 1 إلى n ؛ يجب أن يعاد حساب الثمن عند كل وزنة ، فإذن لا يجب أن يظهر تعريفه في نفس مستوى تعريف النتيجة : لذلك ندخل جدولاً ثانوياً لكل تعريف مكرّر . في المعجم : نسجل بأنه يجب إدخال جدولاً ثانوياً لتعريف الثمن وذلك عن طريق كتابة :

الثمن المتوقع دفعه من قبل الزبون ، بالقروش

prix (entier) pour pesée (صحيح) لـ وزنة)

في المقابل ، فإن تعريف n ليس له علاقة بالوزنة : نعمل على إظهار تعريفه في الجدول الأساسي . المعرف « وزنة Pesée » سيوضع في المعجم ، وهو مشطوب مسبقاً لأنه سبق تعريفه (إنه يساوي 1 ، ومن ثم 2 ، . . . وأخيراً n) .

المعجم	التعريفات
النتيجة = أكتب الثمن لـ وزنة من 1 إلى n	
الثمن (صحيح) لـ وزنة : الثمن المتوقع دفعه من قبل الزبون ، بالقروش	
n - (صحيح) : العدد الإجمالي للوزنات	
n - الوزنة (صحيح)	n وفقاً لمعطية

(كان بمقدورنا تماماً إعطاء قيمة لـ n : $n = 10$. يعتمد هذا الخيار على المحتوى الحقيقي للمسألة)
 بعد أن إكتمل الجدول الأساسي ، فإننا سنكتب الجداول الثانوية ، وذلك بالتجميع في كل منها لكل المعرفين الذين لديهم نفس حقل التعريف .
 سيكون عنوان الجدول الثانوي :
 حقل التعريف ← المعرفين الذي يجب تحديدهم
 (في حالتنا : لـ وزنة ← ثمن) .

المعجم	التعريفات
② - الثمن (صحيح) لـ وزنة : الثمن المتوجب دفعه من قبل الزبون ، بالقروش ② n (صحيح) : العدد الإجمالي للوزنات	① النتيجة = أكتب الثمن لـ وزنة من 1 إلى n
② - الوزنة (صحيح)	③ n وفقاً لمعطية
⑤ الوزن (صحيح) : الوزن بالغرام	④ الثمن = الوزن * السعر الإفرادي $1000 \div$
⑤ السعر الإفرادي (صحيح) : السعر بالقروش للكيلوغرام	⑥ الوزن ، السعر الإفرادي وفقاً لمعطية

ملاحظة : لكي نكون فعلياً دقيقين ، يجب تذييل الثمن ، الوزن والسعر الإفرادي للوزنة بأدلة ، لأنه توجد قيمة في كل خطوة من التكرارية .
 تتم عملية ترتيب التعريفات على كل جدول .
 في لغة الباسكال ، فإن التعريف :

résultat = écrire x pour y de z à u

يُكتب على الشكل التالي :

for y: = z to u do begin

تعريفات
تكرارية

(ترجمة الجدول « لـ y »)
 traduction de la table «pour y»
 writeln (x)

end

(إذا كان الجدول « pour y » فارغاً ، فإنه يمكن إهمال begin و end)
يصبح البرنامج في حالنا هذه :

```

program balance2(input,output);
{ فوترة عدة وزنات على ميزان مسجل }

var prix:integer;    { لكل وزنة ، الثمن المتوجب دفعه من قبل
                    { الزبون ، بالقروش }
    n:integer;        { العدد الإجمالي للوزنات }
    pesee:integer;
    poids:integer;    { لكل وزنة ، الوزن بالغرام }

    prixUnitaire:integer; { لكل وزنة ، الثمن بالقروش للكيلوغرام }

1 → begin read(n);
2 →   for pesee:=1 to n do begin
sous- | 1 →       read(poids,prixUnitaire);
table | 2 →       prix:=poids*prixUnitaire div 1000;
                    writeln(prix)
                    end
end.

```

(sous-table : جدول ثانوي)

عند التنفيذ ، يستلزم هذا البرنامج أولاً قيمة n ، ثم n مرة قيمة الوزن (أي
الوزنة) والسعر الإفرادي (للوزنة) . إذا أعطيناه :

3	500	2000	100	5000	200	1000
1000						يكتب في النهاية
500						
200						

إرجاع الثوابت

إن تحليل المسألة « إجراء n فوترة على ميزان مسجل لسلعة واحدة » ، وبنفس
الطريقة السابقة ، يؤدي الى إمداد البرنامج عند التنفيذ ، n مرة نفس القيمة للسعر
الإفرادي : « السعر الإفرادي وفقاً لمعطية » هو ثابت في التكرارية . من المهم إذن إخراج
هذا التعريف من الجدول الثانوي أي إرجاعه الى الجدول الأساسي :

المعجم		التعريفات
- الثمن (صحيح) لوزنة	3	النتيجة = أكتب الثمن لوزنة من 1 إلى n
- n (صحيح)	1	n وفقاً لمعطية
- لوزنة (صحيح)	2	السعر الإفرادي وفقاً لمعطية
- لوزنة ← سعر		
- الوزن (صحيح)	2	الثمن = الوزن * السعر الإفرادي $1000 \div$
- السعر الإفرادي (صحيح)	1	الوزن وفقاً لمعطية

هذا ما يغير البرنامج :

```

...
begin read(n,prixUnitaire);
  for pesee:=1 to n do begin
    read(poids);
    prix:=poids*prixUnitaire div 1000;
    writeln(prix)
  end
end.

```

ملاحظة : $read(x, y, ..)$ تعني $read(x)$ ؛ $read(y)$ ؛ ...
عند التنفيذ ، إذا كانت المعطيات المقدمة هي :

4 2000 500 200 800 1000

فإن النتيجة المكتوبة تصبح

1000

400

1600

2000

مثال : إطبوع مُربّع الأعداد
المسألة : إطبوع جدولاً لمربّع أوّل 25 عدداً صحيحاً .

التحليل

المعجم	التعريفات
n - (صحيح)	1
$n * n : n$ - مربع (صحيح)	مربع n ، مربع n من 1 إلى 25
$n \leftarrow n$ - مربع	مربع $n * n$

هنا يتعلق المعرف المحدد بالتكرارية (مربع ذي الدليل n) صراحة بدليله ؛ ولا يوجد معطيات (العدد 25 تم تحديده في نص المسألة) . فيما عدا هذه التفاصيل ، فإن التحليل هو نفسه الذي سبق .

البرنامج :

```

program editorCarres(input,output);
{ إطبّع جدولاً لمربع أول 25 عدداً صحيحاً }
var n: integer;
    carre: integer; {n*n}
begin
    for n:=1 to 25 do begin
        carre:=n*n;
        writeln(n,carre)
    end
end. (1)

```

carre = مربع ، éditer = إطبّع ، table = جدول ، entier = صحيح .

ملاحظة : إن جدولة $f(x)$ من أجل x تتغير من A إلى B (أعداد صحيحة) وبخطوات تساوي 1 ، تؤدي إلى برنامج شبيه جداً بهذا البرنامج : نستبدل $n * n$ بـ $f(x)$.

تكرارية للحساب

لنفرض المسألة « إحصاء المبلغ الإجمالي المتوجب دفعه ، بعد عدة وزنات على ميزان مسجل » . لم يعد المقصود كتابة نتيجة عند كل وزنة ، بل كتابة المبلغ الإجمالي ؛ عند كل وزنة جديدة ، يكون المبلغ الإجمالي الجديد هو حاصل جمع المبلغ الإجمالي القديم مع ثمن الوزنة :

(1) في حال كانت طباعة الأرقام الصحيحة تتم بدون تباعد ، فإنه يجب كتابة :
 Writeln (n, ' ', Carre)

$$total_{pesée} = total_{pesée-1} + prix_{pesée}$$

$$(مجموع الورقة = مجموع الورقة - 1 + ثمن الورقة)$$

والنتيجة النهائية المطلوب كتابتها هي المجموع الأخير :
النتيجة = أكتب مجموع

$$مجموع = آخر مجموع ورقة = مجموع ورقة - 1 + ثمن ورقة - 1 + ثمن ورقة من 1 إلى n$$

الأدلة هي غير ضرورية : يكفي وجود إمكانية تفريق الحد الحالي من الحد السابق . من المتفق عليه ، في تعريف ، وضع خط فوق الحد السابق :

$$مجموع = آخر مجموع = مجموع + ثمن - 1 + ثمن من 1 إلى n$$

تُعرّف المتسلسلة « مجموع » بالتثنية إلى الوراثة recurrence (نحسب حدود المتسلسلة بواسطة علاقة تثنية إلى الوراثة (1) ، مثل $u_i = f(u_{i-1})$. لا يمكن إتمام مثل تلك العملية إلا إذا كنا نعرف الحد الأول : مجموع = 0 .

الدليل غير ضروري : يكفي معرفة الحد الأول ، هذا ما نسجله : أول مجموع = 0 .

التعريفات	المعجم
نتيجة = أكتب مجموع	4 - مجموع (صحيح) : متسلسلة المجاميع
المجموع = آخر مجموع = مجموع + ثمن - 1 + ثمن من 1 إلى n	3 - الجزئية ، وزنة تلو وزنة - n (صحيح) : عدد الوزنات
أول مجموع = 0	2 الثمن (صحيح) — -وزنة : متسلسلة
n وفقاً لمعطية	1 الأثمان الواجب دفعها ، بالقروش - الوزنة (صحيح)
الثمن = الوزن * السعر الإفرادي 1000 div	2 - الوزن (صحيح) : الوزن بالغرام
الوزن ، السعر الإفرادي وفقاً لمعطية	1 - السعر الإفرادي (صحيح) : الثمن بالقروش 1 للكيلوغرام

$$u_i = f(u_{i-1}) \quad \{ \text{عن طريق تعريف المتسلسلة بالتثنية إلى الوراثة : } u_0 = a \}$$

$$\text{يمكن حساب } u_2 = f(u_1) = f(f(a)), u_1 = f(a) \text{ ، إلخ } \dots$$

$$\text{مجموع } i = \text{مجموع } i-1 + \text{ثمن } i \text{ ، نحصل على مجموع } i = \text{ثمن } i + \text{ثمن } i-1 + \text{ثمن } i-2 + \dots + \text{ثمن } 1$$

<p>$v = \text{dernier } v = f(\bar{v}) \text{ pour } i \text{ de } \text{départ} \text{ à } \text{arrivée}$ premier $v = \text{expression}$</p>	<p>تكتب التعريفات</p>
---	-----------------------

= expression ، أول = premier ، وصول = arrivée ، انطلاق = départ ، لـ = pour ، آخر = dernier
 تعبير ، جدول = table ، ترجمة = traduction

<p>$v := \text{expression};$ for $i := \text{départ}$ to arrivée do begin traduction de la table « pour i » $v := f(v)$ end</p>	<p>في لغة الباسكال</p>
--	------------------------

بالفعل ، فإن المعرف الموضوع في القسم الأيسر من تعيين (affectation) يتطابق مع
 « القيمة الجديدة » ، بينما يتعلق الأمر في القسم الأيمن بالقيمة القديمة :

$$x = \bar{x} + \dots \Rightarrow x := x + \dots$$

التحليل \Rightarrow باسكال

البرنامج

```

program balance3(input,output);
{calculer le total à payer après plusieurs pesées
sur une balance enregistreuse}
var total:integer; { احسب المبلغ الإجمالي المتوجب دفعه بعد عدة وزنات على ميزان مسجل }
    n:integer;      { متسلسلة المبالغ الإجمالية الجزئية وزنة بعد وزنة }
    prix:integer;   { عدد الوزنات }
    pesee:integer;  { لكل وزنة ، الثمن المتوجب دفعه بالقروش }
    poids:integer;  { لكل وزنة ، بالغم }
    prixUnitaire:integer; { لكل وزنة ، الثمن بالقروش للكيلو }
    
```

1,2 →	begin
3 →	read(n); total:=0;
	for pesee:=1 to n do begin
جدول { 1 →	read(poids,prixUnitaire);
{ 2 →	prix:=poids*prixUnitaire div 1000;
تعريف تكراري →	total:=total+prix
	end;
4 →	writeln(total)
	end.

تعطي المعطيات 10 000 2 1000 5000 2000 2500 النتيجة

مثال : وسط حسابي (moyenne)
 المسألة : احسب الوسط الحسابي لـ n قيمة
 المواصفة : احسب الوسط الحسابي لـ n قيمة (n معطية) :
 وسط حسابي = (مجموع الـ n قيمة) / n ؛ القيم هي حقيقية ،
 (مجموع الـ i قيمة) = (مجموع الـ $i - 1$ قيمة) + (القيمة ذات الترتيب i)

التحليل :

المعجم	التعريفات
② - وسط حسابي (حقيقي) : وسط حسابي	① نتيجة = أكتب وسط حسابي
④ - مجموع (حقيقي) : مجموع الـ n قيمة	③ سطر حسابي = مجموع / n
④ - n (صحيح) : عدد القيم	⑤ مجموع = آخر مجموع + مجموع نمن لـ عدد من 1 إلى n
⑥ - v (حقيقي) لـ عدد : قيمة كل عدد	⑤ أول مجموع = 0
⑥ - عدد (صحيح)	⑦ n وفقاً لمعطية
$v \leftarrow$ عدد	⑧ v وفقاً لمعطية
⑨	

البرنامج :

```

program moyenne(input,output);
{moyenne de n valeurs réelles} { معطى n قيمة حقيقية }
var moyenne:real;                { وسط حسابي لـ n قيمة }
    somme:real;                    { مجموع الـ n قيمة }
    n:integer;                    { عدد القيم }
    v:real;                       { لكل عدد ، قيمته }
    nombre:integer;
begin
    read(n); somme:=0;
    for nombre:=1 to n do begin
        read(v);
        somme:=somme+v
    end;
    moyenne:=somme/n;
    writeln(moyenne)
end.
    
```

moyenne = وسط حسابي ؛ valeur = قيمة ؛ somme = مجموع ؛ nombre = عدد

أمثلة من المعطيات : 0.2 - 0.06 3.14 17.0 - 4 النتيجة : 3.5 -

التكرير للكتابة والحساب

إن المسألة « إحسب المبلغ الإجمالي المتوقع دفعه ، بعد عدة وزونات على ميزان مسجل » تؤدي إلى تعريف نتيجتين ؛ من جهة المجموع النهائي ومن جهة أخرى الثمن الحاصل عن كل وزنة :

نتيجة = نتيجة 1 ، نتيجة 2

نتيجة 1 = أكتب ثمن لـ وزنة من 1 إلى n

نتيجة 2 = أكتب مجموع

سيكون في المعجم ، للنتيجة 1 والنتيجة 2 النوع نص (Text) ، ذلك لكونها قيم خارجية وليست داخلية (مثل حقيقي وصحيح) . إن التعريفات نتيجة 1 ومجموع ، لهم نفس حقل التعريف (لـ وزنة من 1 إلى n) : سيتم دمجهم في البرنامج .

التعريفات	المعجم
① نتيجة = نتيجة 1 ، نتيجة 2	5 - نتيجة 1 (نص)
③ نتيجة 1 = أكتب ثمن لـ وزنة من 1 إلى n	3 - نتيجة 2 (نص)
⑤ نتيجة 2 = أكتب مجموع	4 - الثمن (صحيح) لـ وزنة : متسلسلة الأثمان الواجب دفعها ، بالقروش
⑦ n وفقاً لمعطية	2 - وزنة (صحيح)
⑧ مجموع = آخر مجموع = مجموع + ثمن لـ وزنة من 1 إلى n	3 - n (صحيح) : عدد الوزونات
⑨ أول مجموع = 0	1 - مجموع (صحيح) : متسلسلة المجاميع - الجزئية ، وزنة تلو وزنة
⑩ الثمن = الوزن * السعر الإفرادي 1000 div	2 - الوزن (صحيح) : الوزن بالغرام
⑪ الوزن ، السعر الإفرادي وفقاً لمعطية	1 - السعر الإفرادي (صحيح) : الثمن بالقروش - للكيلوغرام

(12)

بذلك يصبح قسم العبارات في البرنامج :

```
begin
  read(n); total:=0;
  for pesee:=1 to n do begin
    read(poids,prixUnitaire);
    prix:=poids*prixUnitaire div 1000;
    writeln(prix);
    total:=total+prix
  end;
  writeln(total)
end.
```

لا يظهر المعرفان نتيجة 1 ونتيجة 2 في البرنامج ، إنها يتطابقان مع قسمة منطقية للائحة النتائج الغير موجودة في لغة الباسكال .

إذا كانت معطيات البرنامج هي 1000 200 5000 100 2000 500 3
 :نتيجة تصبح 1000
 500
 200
 1700

4.1 - شَرْطِي

عندما تتعلق طريقة حساب قيمة بشرط ، نستعمل تعريفاً شرطياً :

identificateur = expression₁ si condition, expression₂ sinon

(معرف = تعبير₁ إذا شرط ، تعبير₂ وإلا)

إذا تحقق الشرط ، فإن تعريف المعرف يصبح : معرف = تعبير₁ ؛ إذا لم يتحقق الشرط يصبح عندنا : معرف = تعبير₂

هذه هي حالة حساب الساعات الإضافية لعامل بالساعة . نفترض بأن الساعات الإضافية (أكثر من 39 ساعة في الأسبوع) تدفع بنسبة 125% من القيمة الأساسية للساعة ، وبأن حسناً يجري تطبيقه على مجمل الأجر الخام قيمته 4.75% .

المعجم	التعريفات
② - الأجر الصافي (حقيقي) : الأجر الأسبوعي - المستحق	① نتيجة = أكتب الأجر الصافي
④ - الأجر الخام (حقيقي) : الأجر قبل الحسم	③ الأجر الصافي = الأجر الخام - الحسم
④ - الحسم (حقيقي) : 4.75%	⑤ الأجر الخام = الساعات * سعر إذا الساعات = > 39 ، 39 * سعر + (الساعات - 39) * سعر * 1.25 وإلا
⑥ - الساعات (حقيقي) : عدد الساعات المتممة	⑦ الحسم = الأجر الخام * 0.0475
⑥ - سعر (حقيقي) : أجر الساعة	⑧ الساعات وفقاً لمعطية
	⑨ سعر وفقاً لمعطية
	⑩

يترجم الشرط بواسطة مؤثرات مقارنة :

التنويط المتبّع
 = ≠ < ≤ > ≥
 = <> < <= > >=

التنويط في لغة الباسكال

في لغة الباسكال ، يترجم التعريف الشرطي $x = y$ si c, z sinon بالعبارة :

```

if c then begin
  traduction de la table « si c »
  traduction de «  $x = y$  »
end
else begin
  traduction de la table « sinon c »
  traduction de «  $x = z$  »
end

```

(traduction = ترجمة ، table = جدول ، si = إذا ، sinon = وإلا)

ملاحظة : إذا وجدت عبارة واحدة في الشعبة (branche) (then أو else) ، فإنه لا يجب إحاطتها بـ begin و end .

البرنامج :

```

program paye(input,output);
var salaireNet:real; { الدفع الأسبوعي لعامل بالساعة }
    { الأجر الأسبوعي المتوقع }
    salaireBrut:real; { الأجر قبل الحسم }
    retenue:real; { 4.75% من الخام }
    heures:real; { عدد الساعات المشغولة }
    taux:real; { الأجر بالساعة }
begin
  read(taux,heures);
  if heures<=39 then
    salaireBrut:=heures*taux
  else
    salaireBrut:=39*taux+(heures-39)*taux*1.25;
  retenue:=salaireBrut*0.0475;
  salaireNet:=salaireBrut-retenu;
  writeln(salaireNet)
end.

```

(Paye = الدفع ؛ salaire = أجر ؛ retenue = الحسم ؛ brut = خام ؛ Net = صافي ؛ taux = سعر ، heure = ساعة)

النتيجة 815.24	معطيات : 30,0 28.53
النتيجة 1739.19	معطيات : 59.0 28.53

مثال : معادلة من الدرجة الثانية

حل معادلة من الدرجة الثانية $ax^2 + bx + c = 0$ ، حيث $a \neq 0$

بعد مراجعة كتابات متخصصة ، يتبين بأن لهذه المعادلة جذرين حقيقيين x_1 و x_2 إذا كانت الكمية $d = b^2 - 4ac$ هي موجبة أو تساوي صفر ، ولها جذرين عُقديين $R - iS$ و $R + iS$ (Complex) إذا كانت d سالبة :

résultat = écrire x_1, x_2 si $d \geq 0$, écrire $R, ' + i', S, R, ' - i', S$ sinon

(écrire = أكتب ، résultat = نتيجة ، si إذا ، sinon وإلا)

إن التنويط '+' يعني بأنه يجب كتابة السمتين '+' و 'i' كما هما . إن شكل النتيجة سيكون : $2.0 - i4.5$ ؛ إن سلسال السمات المنوّط بين علامات حذف يكتب كما هو .

سيتم كتابة x_1 و x_2 في المعجم مع حقل تعريف ' $d \geq 0$ ' ؛ R و S مع الحقل ' $d < 0$ '

المعجم	التعريفات
② x_1 - (حقيقي) : $d \geq 0$: جذر حقيقي	① نتيجة = أكتب 'جذرين حقيقيين : x_1, x_2 ، و' x_2 إذا $d \geq 0$ ، أكتب 'جذرين عُقديين : $R, ' + i', S, R, ' - i', S$ ، وإلا
② x_2 - (حقيقي) : $d \geq 0$: جذر حقيقي	③ $d = b^2 - 4 * a * c$
② R - (حقيقي) : $d \geq 0$: القسم الحقيقي	⑤ a, b, c وفقاً لمعطية
② S - (حقيقي) : $d \geq 0$: القسم التخيلي من الجذور العُقدية	
② d - (حقيقي) : مُميز (discriminant)	
④ a - (حقيقي) : مُعامل x^2	
④ a - (حقيقي) : مُعامل x^1	
④ c - (حقيقي) : مُعامل x^0	
$d \geq 0 \rightarrow x_1, x_2$	⑥ $x_1 = (-b + \sqrt{d}) / (2 * a)$
	⑦ $x_2 = (-b - \sqrt{d}) / (2 * a)$
$d < 0 \rightarrow R, S$	⑧ $R = -b / (2 * a)$
	⑨ $S = \sqrt{-d} / (2 * a)$
⑩	

تنوط العملية \sqrt{d} في لغة الباسكال كما يلي : Sqrt (d)

البرنامج

```

program secondDegre(input,output);
{ حل  $ax^2 + bx + c = 0$  }
var x1,x2:real;      { إذا  $d \geq 0$  جذور حقيقية }
    R,S:real;        { إذا  $d < 0$  ، الأقسام الحقيقية والتخيلية للجذور العقدية }

    d:real;           { مميز }
    a,b,c:real;      { معاملات }
begin
    read(a,b,c);
    d:=b*b-4*a*c;
    if d>=0 then begin { جذرين عقديين }
        x1:=(-b+sqrt(d))/(2*a);
        x2:=(-b-sqrt(d))/(2*a);
        writeln('deux racines reelles:');
        writeln(x1,' et ',x2)
    end
    else begin { جذرين حقيقيين }
        R:=-b/(2*a);
        S:=sqrt(-d)/(2*a);
        writeln('deux racines complexes:');
        writeln(R,'+',S,'i' et R,'-',S,'i')
    end
end.

```

. (حقيقي = réelle ، عقدي = complexe ، جذر = racine ، الدرجة الثانية = second degré)

معطيات : 4.0 - 4.0 2.0

النتيجة : جذرين عقديين $1.0 + 1.0i$ و $1.0 - 1.0i$

معطيات : 0.0 1.5 1.0

النتيجة : جذرين حقيقيين 0.0 و -1.5

الشرطية والتكرارية

في متسلسلة من 10 معطيات صحيحة ، نودّ عدّ تلك التي هي موجبة أو مساوية لصفر ، وكذلك تلك السالبة .

المعجم	التعريفات
pos - (صحيح) : عدد المعطيات الموجبة أو المساوية لصفر	3 نتيجة = أكتب 'موجبة' = ' ، pos ، ' سالبة' ، neg
neg - (صحيح) : عدد المعطيات السالبة	2 آخر pos = pos آخر x + pos = pos من 1 إلى 10
x - (صحيح) : إذا كانت المعطية موجبة أو مساوية لصفر	1 أول pos = 0
i - (صحيح)	3 pos - 10 = neg
$x \leftarrow i$	
v - (صحيح) : معطية مقروءة	2 $x = 1 \text{ si } v \geq 0, 0 \text{ sinon}$
	1 وفقاً لمعطية v

البرنامج

```

program positif(input,output);
{ عد القيم الموجبة أو المساوية لصفر والقيم السالبة ، في متسلسلة من 10 معطيات }
var pos:integer; { عدد المعطيات الـ >= 0 }
    neg:integer; { عدد المعطيات الـ < 0 }
    x:integer; { لكل معطية 1 إذا موجبة ، وإلا 0 }

    i:integer;
    v:integer; { لكل معطية = القيمة المقروءة }
begin
    pos:=0;
    for i:=1 to 10 do begin
        read(v);
        if v>=0 then x:=1 else x:=0;
        pos:=pos+x;
    end;
    neg:=10-pos;
    writeln('positives=',pos,' negatives=',neg)
end.

```

(positives = موجب ، negatives = سالب)

معطيات : 111 0 -4 1 2 0 3 -4 -7 1

نتيجة : موجب = 7 ، سالب = 3

5.1 تكرارية مع توقف

إن توقف التكرارية يمكن أن يحدد :

- صراحة : id = تعريف لـ i من « إنطلاق » إلى « وصول » إذا عرفنا مسبقاً عدد خطوات التكرارية .

- بواسطة شرط ؛ في هذه الحالة فإن خطوات التكرارية تتم عندما :
- يتحقق الشرط :

id = تعريف لـ i من « إنطلاق » طالما « شرط »
إن آخر قيمة ناتجة عن هذه العملية ، أي أول قيمة غير محققة للشرط ، سوف تُقضى
عن التكرارية (EXCLUDE)
- الشرط غير متوافر

id = تعريف لـ i من « إنطلاق » متضمناً (inclus) « شرط »
إن آخر قيمة ناتجة هي داخلية في التكرارية .

تكرارية مع إقصاء (أو إقتصار) (Itération avec exclusion)
إن إرادة الإقصاء من التكرارية ، للحدّ الأخير الناتج تعني بأن المتتالية المكررة
سوف تنتهي بانتاج حدّ الذي رُبما إستُخدِم في الخطوة التالية :

<p>إنتاج الحدّ الأولي ، من ثمّ إذا توفر الشرط العمل على</p> <p>- إستعمال الحدّ</p> <p>- إنتاج الحدّ التالي .</p> <p>- من ثمّ إعادة الكرة</p>	<p>إقصاء ← طالما ←</p> <p>(tant que)</p>
--	--

إن التعبير « إنتاج حدّ » يعني هنا حساب الحدّ الفعلي ، دليله في المتسلسلة ، وقيمة
الشرط المتوافق ؛ عند الترجمة إلى الباسكال ، فإن ، حساب الشرط ، إذا كان سهلاً ،
يمكن أن يتم في لحظة الإختبار ويمكن إلغاء عملية حساب الدليل إذا لم يكن الدليل مستعملاً
صراحة .

معرف = تعريف لـ i من « إنطلاق » طالما « شرط »
يمكن إعادة كتابة تلك العبارة كما يلي :

(إنطلاق)
i: = départ;
calcul du terme initial (d'indice départ) (حساب الحدّ الأولي)
c: = condition (sur le terme initial) (شرط على الحدّ الأولي)
while c do begin
(ترجمة إستعمال الحد) traduction de l'utilisation du terme (= تعريف (1))
i: = i + 1;
(ترجمة إنتاج الحد) traduction de la production du terme (= جدول ثانوي
c: = condition (sur le terme produit) (شرط على الحدّ المنتج)
end

(1) إلا إذا كان المقصود هو تعريف مكرر ، الذي يؤخذ إذا في هذه الحالة كقسم من الجدول الثانوي (إنتاج) .

(إن المَعْرِفَ المُمَثِّلَ للشرط «C» هنا ، سيتم التصريح عنه في الباسكال مع النوع «البولي»)

مثال : مجموع
لنفرض المطلوب حساب مجموع عدة قيم صحيحة موجبة ؛ لا نعرف عددها ،
لكنها متبوعة بالقيمة «-1» .
يتعلق الأمر إذن بتكرارية مع شرط توقف ، الحد الأخير هو مقصي عن الحساب .

المعجم	التعريفات
- مجموع (Somme) (صحيح)	3 نتيجة = أكتب مجموع
- V (صحيح) \downarrow n : متسلسلة من القيم	2 مجموع = آخر مجموع = مجموع $v \downarrow n$ من 1
- n (صحيح)	1 طالما $v > -1$
	أول مجموع = 0
	$v \leftarrow n \downarrow$ 1 وفقاً لمعطية

كون الشرط سهل التعبير ، فسيتم ترجمته مباشرة ؛ وبما أن الدليل غير ظاهر
صراحة ، فسيتم إلغاءه :
البرنامج

```

program somme(input,output);
{ مجموع القيم المقروءة ، متبوعة بـ -1 }
var somme:integer;
    v:integer;      { لكل معطية ، قيمتها }
begin
    somme:=0;
    read(v);
    while v<>-1 do begin
        somme:=somme+v;
        read(v)
    end;
    writeln(somme)
end.

```

تكرارية مع تضمين (Itération avec inclusion)
يتم إستعمال الحد الأخير الناتج :

<p>التدميث عند الإقتضاء ، من ثم</p> <p>- إنتاج حدّ</p> <p>- إستعماله</p> <p>- من ثم إعادة الكرة إذا لم يتوفر الشرط</p>	<p>تضمين ← متضمناً (inclus) ←</p>
--	-----------------------------------

معرف = تعريف لـ i من إنطلاق متضمناً شرط
يمكن إعادة كتابة تلك العبارة كما يلي :

```

i := départ - 1;
repeat
    i := i + 1;
    ( = إنتاج )      traduction de la table « pour i »      ( ترجمة الجدول لـ i )
    ( = إستعمال )    traduction de la définition          ( ترجمة التعريف )
    c := condition    ( شرط )
until c

```

حيث ان فتح المزدوجين begin-end لا لزوم له
ملاحظة : هذا الشكل من التكرارية يستعمل خاصة لجمع المتسلسلات .

مثال : نيوتن Newton

إن الجذر التربيعي لعدد موجب a هو أيضاً نهاية المتسلسلة المكررة

$$u_i = (u_{i-1} + u_{i-1}) / 2 \quad (\text{صيغة نيوتن}) .$$

بالفعل ، إذا سمينا نهاية المتسلسلة ، فإن 1 تخضع للمعادلة $1 = (1 + a/1)/2$
أي $1^2 = a$. لكي نحصل على 1 مع دقة إسيلون (precision epsilon) معطية ، يجب
توقيف التكرار للقيمة x للدليل بشكل أن $|a - U^2_x| / a$ تصبح أصغر من إسيليون
(القيمة الأولية u_0 هي غير ذات أهمية كبرى) :

معجم	تعريفات
u - (حقيقي): متسلسلة التقريبات	نتيجة = أكتب u
a - (حقيقي): العدد الذي نبحث عن جذره	آخر u = $(\bar{u} + a/\bar{u})/2$ من k لـ 1
k - (صحيح)	متضمناً $ a - u /a < \epsilon$
- إبسيلون (حقيقي): دقة	أول $1 = u$
	a وفقاً لمعطية
	إبسيلون = 0.00001

تنوّط القيمة المطلقة في لغة الباسكال abs .

البرنامج

```

program Newton(input,output);
{ حساب الجذر التربيعي ، على طريقة نيوتن }
var u:real; { متسلسلة التقريبات }
    a:real; { العدد الذي نبحث عن جذره }
    epsilon:real; { دقة }
begin
    epsilon:=0.0001; read(a); u:=1;
    repeat
        u:=(u+a/u)/2
    until (abs(a-u*u)/a)<epsilon;
    writeln(u)
end.

```

إذا أردنا الحصول في نفس الوقت على الجذر وعدد التكريرات ، فإنه يكفي تغيير التعريف 5 إلى أكتب k, u لكي يجب عندئذٍ وصف الحسابات المجراة على k في البرنامج :

```

begin
    epsilon:=0.0001; read(a); u:=1; k:=0;
    repeat
        k:=k+1;
        u:=(u+a/u)/2
    until (abs(a-u*u)/a)<epsilon;
    writeln(k,u)
end.

```

دون أن ننسى التصريح k : integer

التكريرات

عدد الخطوات المعلومة		
بواسطة شرط		صراحة
الحد الأخير مقصي	الحد الأخير ضمناً	pour i de d à a
pour i de d tant que c (i)	pour i de d inclus c (i)	
while	repeat	for
0 مرة	1 مرة واحدة	0 مرة
قبل التكرارية	بعد التكرارية	اختبار التوقف

عبارة صريحة

باسكال

قد تمت التكرارية
على الأقل

مثال : إحصائيات

في جملة دراسات إحصائية ، يوجد حاجة لمعرفة عدد الفراغات (تباعد) في نص
مكوّن على الشكل التالي :

1 - يوجد ثلاث فقرات ، مؤلفة من جمل ، تنتهي الفقرة بجملة لا تحتوي على تباعد ومؤلفة
جزءاً من الفقرة .

2 - تتكون كل جملة من عدة سمات بما فيها التباعد وتنتهي بالسمة . التي لا تدخل ضمن
نطاق الجملة .

نودّ إذن الحصول بالنسبة لكل فقرة على متسلسلة اعداد التباعدات والسمات في كل
جملة ، مفصولة بخط وصل (-)
مثلاً في حال المعطية التالية :

EN FAIT. IL Y A TIL. UN. RADEAU. AVEC UN S. DEDANS

فإن النتيجة تكون : 1-7, 4-11, 0-2, 0-6, 2-9, 0-6

ملاحظة : يَنُوط النوع سمة في لغة الباسكال «char» ؛ يمكن قراءة وكتابة سمات
كما يمكن مقارنتها في ما بينها .

تعريفات		معجم
نتيجة = عالج لـ فقرة من 1 إلى 3		1 - عالج (نص) لـ فقرة - فقرة (صحيح)
		لـ فقرة ← عالج
عالج = أكتب nb Car, ' ', nb Espaces لـ جملة من 1 متضمنًا nb Espaces = 0		1 - nb Espaces (صحيح) لـ جملة : عدد التباعد في جملة - nb Car (صحيح) لـ جملة : عدد السمات في جملة - جملة (صحيح)
		لـ جملة ← nb Car, nb Espaces
$\overline{\text{nb Espaces}} = \text{nb Espaces آخر}$ $\overline{\text{carac} < > ' ' \text{ لـ } x \text{ من } 1 \text{ طالما } ' ' < > \text{ carac}}$ $\text{nb Car آخر} = \text{nb Car} + \text{nb Car}$ $\text{لـ } x \text{ من } 1 \text{ طالما } ' ' < > \text{ carac}$ $0 = \text{nb Espaces أول}$ $0 = \text{nb car أول}$		3 - x (صحيح) لـ c: 1 إذا كانت هي تباعد
		3 - carac (سمة) لـ c: سمة ذات الرقم c
		1 - c (صحيح)
		1
		لـ c ← carac, x
x = 1 إذا ' ' carac = 0 وإلا carac وفقاً لمعطية		2 1

إن المسألة التي فيها نستبدل سمة بعدد صحيح

نقطة بـ 1 -

تباعد بـ 0

تبدو أسهل . لكن لا شيء يفيد ، فتعقيد التحليل يبقى كما هو (مع الأخذ بعين الاعتبار أن معالجة القيم الرقمية يمكن أن تظهر مألوفة أكثر) .

البرنامج

```

program statistiques(input,output);
{ عدد التباعدات في نص مؤلف من ثلاث فقرات ، مؤلفة من جمل تنتهي الفقرة بجمله }
{ دون تباعد . تنتهي الجملة بنقطة ، التي لا تدخل ضمن الجملة . يطلب عدد نسب التباعدات ، جملة بجملة }
var paragraphe,phrase,c:integer; { دلائل }
    nbEspaces,nbCar:integer; { عدد التباعدات ، السمات في كل جملة }
    x:integer; { لكل سمة ، 1 إذا كان تباعداً }

    caract:char; { لكل سمة ، قيمتها }
begin
  for paragraphe:=1 to 3 do
  PARA( فقرة )
  repeat
    nbCar:=0; nbEspaces:=0;
    read(caract);
    if caract=' ' then x:=1 else x:=0;
    while caract<>'.' do begin
      nbCar:=nbCar+1; nbEspaces:=
        nbEspaces+x;
      read(caract);
      if caract=' ' then x:=1 else x:=0
    end;
    writeln(nbEspaces,' - ',nbCar)
  until nbEspaces=0
  end.

```

إنتاج

إستعمال (تضمين)

أولاً

جملة

إستعمال

إنتاج (إقصاء)

إستعمال

Statistiques : إحصائيات ؛ compteur = عد ؛ espaces : تباعد ؛ texte : نص ؛ paragraphes : فقرات ؛ phrase : جملة ؛ point : نقطة .

6.1 - جداول (Tables)

في المثال السابق ، كان بالإمكان دمج التكراريتين (المعرفتان لـ nb Espaces و nb Car) ؛ كل معطية مقروءة تم في الحال إستعمالها في معالجتين مستقلتين . حالات أخرى يمكن كذلك أن تظهر :

- لوضع فاتورة لعدة سلع مختلفة ، فسنراجع التعرفة ، الممثلة بواسطة جدول : دالة ذات قيم حقيقية (الأسعار) لتغير صحيح (إسنادات السلع) ؛
- لكي نقدّر تشتت القيم في متسلسلة ، نحسب الإنحراف المعياري (écart type) ، هذا ما يستتبع إستعمال المتسلسلة مرتين ؛ لذا نحفظها في جدول : تركيب أسس الحفظ ومن ثم إستعماله لمرات عدة .
- لكي نحسب التكرّر الخاص بسمات في نص ، نؤلف متسلسلة جداول لقياس المسار في النص : تركيب أسس الحفظ بواسطة التكرارية ، من ثم إستعماله لمرة واحدة فقط ؛
- لكي نبرمج عمليات خاصة بالمصفوفات ، نعرّف جداولاً ذات عدة أبعاد ؛

- لكي نبرمج جدول تقرير ؛ أي صيغة شرطية ذات عدة خيارات ، فإنه يمكن إستعمال ، تبعاً للحالة ، جدولاً للصيغ الشرطية ، أو كذلك عبارة خيارات .
في كل هذه الأمثلة ، إن الإنطلاق من النتيجة يسهل التحليل .

جدول : دالة لتغيير صحيح

لكي نضع فاتورة ، فإننا نجمع أثمان كل سلعة مطلوبة :

$$\text{total} = \sum_p \text{prixUnitaire}_p \times \text{quantité}_p$$

(total : مجموع ؛ Prix unitaire : سعر إفرادي ؛ quantité : كمية) .

حيث أن الـ P هي اسنادات السلع .

في الحالة التي يكون فيها الإسناد عدد صحيح ، موجود بين 1 و10 مثلاً ، فإن سلسلة الـ 10 أسعار إفرادية توصف في المعجم بـ « سعر إفرادي [10...1] (حقيقي) » ويكتب السعر الإفرادي الخاص بالعنصر P « سعر إفرادي [P] » .

بالفعل فإن المراد من سلسلة الـ 10 قيم حقيقية سعر إفرادي [1] ، سعر إفرادي [2] ، ... ، سعر إفرادي [10] هو إمكانية الكتابة السهلة (معالجة الـ 10 معرفين سعر إفرادي 1 ، سعر إفرادي 2 ، إلخ ... هو أمر صعب ومتعب) .

إن التنويط المقابل لذلك في لغة الباسكال هو :

prixUnitaire : array [1..10] of real;
et prixUnitaire [p]

إذا تفق على أن المشتريات تنتهي بالمشتري « تنكة » للسلعة 0 ، فإننا نحصل على :

تعريفات	معجم
نتيجة = أكتب مجموع	- مجموع (حقيقي) : الثمن المتوقع دفعه
مجموع = آخر مجموع = مجموع + ثمن لـ a	- الثمن (حقيقي) لـ a : الثمن الإجمالي
من 1 طالما سلعة < > 0	للـ a _{ième} مُشترى
أول مجموع = 0.0	- a (صحيح)
	- سلعة (صحيح) لـ a : اسناد الـ a _{ième} مشتري
	لـ a ← ثمن ، سلعة
ثمن = كمية * سعر افرادي [سلعة]	- كمية (صحيح) : عدد الأشياء المشتراة

إذا كان مكان الـ « سعر إفرادي [سلعة] » هو فعلاً في الجدول الثانوي ، فإن مكان سلسلة الأسعار الإفرادية « سعر إفرادي [1 ... 10] » هو في الجدول الأساسي : إن « سلعة » هي التي تتعلق بالتكرارية وليس « سعر إفرادي » . إذن نضيف على معجم الجدول الأساسي : سعر إفرادي [1 10] (حقيقي) : تعرفه .
ونتهي الجدول الثانوي بـ :

كمية ، سلعة وفقاً لمعطية

أخيراً يبقى أن نعرف السعر الإفرادي في الجدول الأساسي :

سعر إفرادي = سعر إفرادي [i] وفقاً
لمعطية لـ i من 1 إلى 10

البرنامج

```
program facture(input,output);
{ فوترة مُشترى عدة سلع ، مع إستعمال تعرفه }
var total:real;      { الثمن المتوجب دفعه }
    prix:real;       { لكل مشترى ، ثمنه }
    a,i:integer;
    produit, quantite:integer; { لكل مشترى ، إسناده في التعرفة ، وعدده }
    prixUnitaire:array[1..100] of real; { تعرفه }
begin
    for i:=1 to 10 do read(prixUnitaire[i]);
    total:=0.0;
    read(quantite,produit);
    while produit<>0 do begin
        prix:=quantite*prixUnitaire[produit];
        total:=total+prix;
        read(quantite,produit)
    end;
    writeln(total)
end.
```

(achat : مشترى ؛ produit : سلعة ؛ tarif : تعرفه ؛ quantite : كمية ؛ total : مجموع)

إذا كان لدينا المعطيات :

5 2 3 8 1 1 2 8 9 4 1 2 2 4 1 9 5 5 0 0

تكون النتيجة : 32

جدول : إستعمال

على متسلسلة S من N قيمة ، نعرف :

$$m = \frac{\sum_{i=1}^N S_i}{N} \quad \text{الوسط الحسابي}$$

الانحراف المعياري ، تقدير انحراف القيم من وسطهم الحسابي

$$e = \sqrt{\frac{\sum_{i=1}^N (S_i - m)^2}{N}}$$

سنستعمل إذن الوسط الحسابي m لحساب الانحراف المعياري : S ستكون جدولاً .

معجم	تعريفات
m - (حقيقي) : وسط حسابي	نتيجة = أكتب e, m
e - (حقيقي) : انحراف معياري	$m = S1 / N$
S1 - (حقيقي) : مجموع القيم	$e = \text{Sqrt}(s2 / N)$
N - (صحيح) : عدد القيم	$S1 = \text{dernier } S1 = \overline{S1} + S[i] \text{ pour } i$
	de 1 à N
S2 - (حقيقي) : $\sum (S_i - m)^2$	أول S1 = 0.0
S [1...N] - (حقيقي) : جدول الـ N قيمة	N وفقاً لمعطية
i - (صحيح)	$S2 = \text{dernier } S2 = \overline{S2} + (S[j] - m) * (S[j] - m) \text{ pour } j \text{ de } 1 \text{ à } N$
j - (صحيح)	أول S2 = 0.0
K - (صحيح)	S [K] = S وفقاً لمعطية لـ K من 1 إلى N

البرنامج

```

program ecartType(input,output):
{ حساب الوسط الحسابي والانحراف المعياري }
var m:real;      { وسط حسابي }
    e:real;      { انحراف معياري }
    S1:real;     { مجموع القيم }
    S2:real;     { مجموع مربعات الانحرافات }
    N:integer;   { عدد القيم }
    S:array [1..1000] of real; { جدول الـ N قيمة }
    i,j,k:integer; { قيمة مُكَبَّر أبعادها }

```



```

begin
    تركيب → read(N); for k:=1 to N do read(S[k]);
    استعمال → S1:=0.0; for i:=1 to N do S1:=S1+S[i]; m:=S1/N;
    استعمال → S2:=0.0; for j:=1 to N do S2:=S2+(S[j]-m)*(S[j]-m);
               e:=sqrt(S2/N);
               writeln(m,e)
end.

```

(écart type = انحراف معياري)

في لغة الباسكال ، تكون الحدود المصرّحة لدليل الجدول من الثوابت ؛ من هنا تكبير أبعاد الجدول S .

جدول : تكوين

في نص مؤلف من السمات '0', '1', ..., '9' فقط ومنتهي بنقطة ، نسعى لحساب التكرار الخاص بالسمات - الأرقام .

ملاحظة : 1 . يوجد على النوع سمة (char) علاقة ترتيب relation d'ordre :

$$'0' < '1' < \dots < '9'$$

ودالتين succ و pred تسمحان بالانتقال من سمة إلى أخرى أكبر مباشرة (أو أصغر) :

$$succ('0') = '1' \quad pred('5') = '4'$$

2- إن النوع سمة ، كما النوع صحيح ، هو نوع ترتيبى ، يُمكن :

- تدليل الجدول بعدد صحيح ، أو بسمة .
 - كتابة تكرارية تعمل على فترة من الأعداد الصحيحة أو من السمات (أنظر الفصل 2) .
- المطلوب هنا تكوين متسلسلة من الجداول لقياس مسار النص :

معجم	تعريفات
c - (سمة)	6 نتيجة = أكتب c ، c ∈ F [c] من '0' إلى '9'
F ['0' ... '9'] - (حقيقي) : تردد	5 $F = F [x] = N [x] / qté$ pour x de '0' à '9'
N ['0' ... '9'] - (صحيح) : عدّ	2 $N = dernier N ['0' .. pred(i)] = \overline{N} ['0' .. pred(i)] =$ $N [i] = \overline{N} [i] + 1$ $N [succ(i) .. '9'] = \overline{N} [succ(i) .. '9']$ pour carac de 1 tant que i < > '9'
x - (سمة)	
qté - (صحيح) : عدد السمات	
i - (سمة) : سمة متداولة	
carac - (صحيح) : دليل السمة المتداولة	1 $premier N = N [j] = 0$ pour j de '0' à '9'
j - (سمة)	4 $qté = dernier qté = \overline{qté} + N [k]$ pour k de '0' à '9'
k - (سمة)	3 $0 = qté أول$
$i \leftarrow carac$	
1	
i وفقاً لمعطية	

هذه المسألة تؤدي إلى مظاهر عمليات على جداول كاملة (تعيين ، قسمة بقيمة في القاعدة 5 الجمع في القاعدة 2) ؛ إن استعمال العمليات على جداول كاملة هي أداة التحليل العادي ؛ هنا ، نصفهم لكي نقرب من التنويطات المستعملة في لغة الباسكال .

البرنامج

```

program frequencies(input,output);
{ حساب تردد سمات ( أرقام ) في نص منتهي بنقطة }

var c,j,k,x:char;
    F:array['0'..'9'] of real;      { تردد }
    N:array['0'..'9'] of integer;    { عدّ }
    qte:integer;                     { عدد السمات }
    i:char;                           { سمة متداولة }
    carac:integer;                   { ترتيب السمة المتداولة }

begin
  for j:='0' to '9' do N[j]:=0;
  read(1);
  while i<>'.' do begin
    N[i]:=n[i]+1;
    read(1)
  end
end

```

```
end;
qte:=0; for k:='0' to '9' do qte:=qte+n[k];
for x:='0' to '9' do F[x]:=N[x]/qte;
for c:='0' to '9' do writeln(c,F[c])
end.
```

ملاحظة : إن التكريرات المستعملة على x و c يمكن لها أن تندمج وتصبح :

```
for x: = '0' to '9' do writeln (x, N [x]/qte)
```

إذا كانت المعطيات هي :

3 1 3 7 9 1 3 4 1 1.

1	0.4
2	0.0
3	0.3
4	0.1

فإن النتيجة تصبح

إلخ

إن المسألة المحصول عليها باستبدال سمة بـ رقم
نقطة بـ عدد ما

يمكن أن تظهر أكثر سهولة (يجب التجريب) ، لكنها فعلياً بنفس مستوى التعقيد .

مصفوف (Matrice)

نسمي بالمصفوف ، الجدول ذي البعدين ، أي المدلل بدليلين . مثلاً جدول
العلامات على 20 لـ x في y مادة هو مصفوف :

مادة	y	j	2	1	
	علامة 1,y	علامة 1,2	علامة 1,1	1
	علامة 2,y	علامة 2,2	علامة 2,1	2
	i
	علامة x,y	علامة x,2	علامة x,1	x
				تلميذ

تنويطات في التحليل

note [1... x, 1... y] (حقيقي)

note [i, j]

وفي لغة الباسكال

array [1...x, 1... y] of real

note [i, j]

(note = علامة)

حيث أن x و y هما ثوابت . فيما يلي سنفترض وجود تلميذين وثلاثة مواد . لكي نستثمر هذا المصفوف ، يوجد عدة عمليات ممكنة :
- إن حساب الوسط الحسابي للقيم على السطر i ، يعني حساب الوسط الحسابي للتلميذ i :

$$\text{moyenneElève}_i = \left(\sum_{j=1}^y \text{note}_{i,j} \right) / y$$

(moyenne = وسط حسابي ؛ élève = تلميذ) *

تعريفات		معجم
نتيجة = أكتب y / Si لـ i من 1 إلى x	4	Si - (حقيقي) لـ i : مجموع التلميذ i
3 = y	2	y - (صحيح) : (3) عدد المواد
2 = x	1	i - (صحيح)
		x - (صحيح) : (2) عدد التلاميذ
علامة وفقاً لمعطية	3	note [1...x, 1...y] - (حقيقي)
Si = Si آخر = Si + علامة [i, j] لـ j من 1 إلى y أول Si = 0.0		<div style="border-left: 1px solid black; padding-left: 10px;"> <div style="border-bottom: 1px solid black; margin-bottom: 5px;"> <i>S ← i</i> لـ </div> <div style="display: flex; justify-content: space-between;"> <div style="width: 40%;"> غير متغير صاعد </div> <div style="width: 50%;"> 2 1 </div> </div> </div>
		j - (صحيح)

يجب على عملية قراءة المصفوف أن تكون مشروحة كاملاً في لغة الباسكال .
x و y تلعبان دور الثوابت ؛ سنستعملهما كما هما في البرنامج وذلك بالتصريح عنها
const y = 3 ; x = 2

قبل تصريح المتغيرات .

البرنامج

```

program moyenneEleves(input,output);
{ معدلات x تلميذ في امتحان من y مادة }
const y=3; { مواد } x=2; { تلامذة }
var Si:real; { لكل طالب ، مجموع }
    i,j,il,jl:integer;
    note:array[1..x,1..y] of real;
begin
    for il:=1 to x do
        for jl:=1 to y do
            read (note[i1,j1]);
        for i:=1 to x do begin
            Si:=0.0; for j:=1 to y do Si:=Si+note[i,j];
            writeln(Si/y)
        end
    end
end.

```

إذا كانت المعطيات هي :

11 16 6 9 12 6

فالناتجة تصبح : 9 11

- إن حساب الوسط الحسابي للقيم على العمود z ، يعني حساب الوسط الحسابي المحصل عليه في مادة من قبل مجموع التلاميذ :

$$\text{moyenne Matière}_j = \left(\sum_{i=1}^x \text{note}_{i,j} \right) / x$$

(مادة = Matière)

ناتجة = أكتب x / Sj - j من 1 إلى y

Sj = آخر Sj = علامة [i , j] - i من 1 إلى x

```

...
for j:=1 to y do begin
    Sj:=0.0; for i:=1 to x do Sj:=Sj+note[i,j];
    writeln(Sj/x)
end
...

```

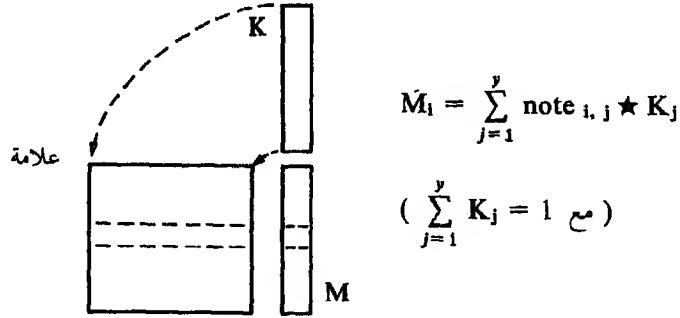
مع المعطيات

11 16 6 9 12 6

الناتجة 10 14 6

- إن القيام بضرب المصفوف علامة بمتجه (Vecteur) (جدول ذي بعد واحد) ذي y

قيمة ، يعني حساب الوسائط الحسابية للتلاميذ آخذين بعين الاعتبار معاملات المواد :



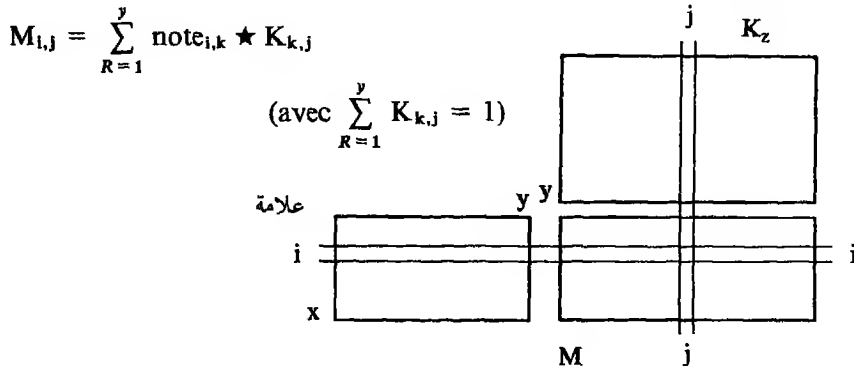
نتيجة = أكتب M لـ i من 1 إلى x
 K وفقاً لمعطية

...
 $K[1..y]$ (حقيقي)
 ...

.....
 $\overline{M} = M \text{ آخر} = M + \text{علامة}[i,j] \star K[j]$
 لـ j من 1 إلى y

...
 for i:=1 to x do begin
 M:=0.0; for j:=1 to y do M:=M+note[i,j]*V[j];
 ...

- إن القيام بضرب المصفوف علامة بمصفوف ذي y سطر و z عمود ، يعني حساب الوسائط الحسابية لكل تلميذ في z تلاعب بالمعاملات :



```

for i:=1 to x do
  for j:=1 to z do begin
    M[i,j]=0.0;
    for k:=1 to y do
      M[i,j]:=M[i,j]+note[i,k]*V[k,j]
    end
  end
end

```

خيارات

يمكن أن يكتب التعريف الشرطي بشكل جدول تقرير ذي خيارين :

	شرط	متوفر	عاجز
$x = y$ si condition, z sinon	\rightarrow	\downarrow	\downarrow
$x =$	\leftarrow	y	z

يمكن بسهولة تخيل قرار بثلاثة خيارات :

b مقارنة مع a	$a < b$	$a = b$	$a > b$
$x =$	e1	e2	e3

أو بـ n خيار :

إذا سمة c	\rightarrow	+	-	★	/	...
r إذن =	\leftarrow	$a + b$	$a - b$	$a \star b$	a/b	

نكتب التعريفات المقابلة كما يلي :

$$x = e1 \text{ si } a < b, e2 \text{ si } a = b, e3 \text{ si } a > b$$

$$r = a + b \text{ si } c = '+', a - b \text{ si } c = '-', a \star b \text{ si } c = '\star', a/b \text{ si } c = '/'$$

حيث أن سرد الشروط المتتالية يجب أن يشمل كل الإمكانيات ، لأنه لا يوجد قرار « وإلا »

في لغة الباسكال ، تكون الترجمة دائماً ممكنة بواسطة إختبارات شلشلية

```

if a < b then x:=e1
else if a=b then x:=e2
else if a>b then x:=e3
else ...

```

عندما تدور مختلف الشروط حول تعداد قيم تعبير من النوع المرتب (type ordinal)
(صحيح ، بولي ، سمة) ، فإننا نكتب :

```

case expression of
  valeur1 : énoncé1;
  valeur2 : énoncé2;
  ...
end

```

(expression = تعبير ؛ valeur = قيمة ؛ énoncé = عبارة)

لنفرض مثلاً أننا نريد كتابة برنامج محاسب صغير مَتمم للأربع عمليات على أعداد حقيقية .

معجم	تعريفات
r- : النتيجة المحسوبة	3
a- (حقيقي) : متأثر 1	2
b- (حقيقي) : متأثر 2	1
c (سمة) : رمز حسابي	
	r = a + b Si c = '+', a - b Si c = '-', a * b Si c = '*', a / b Si c = '/'
	نتيجة = أكتب r c, b, a وفقاً لمعطية

البرنامج

```

program calculette(input,output);
{ الأربع عمليات على اعداد حقيقية }
var r,a,b:real; { النتيجة ، المتأثرين }
    c:char; { المؤثر : + - * / أو }
begin
  read(a,c,b);
  case c of
    '+': r:=a+b;
    '-': r:=a-b;
    '*': r:=a*b;
    '/': r:=a/b;
  end;
  writeln(r)
end.

```

(calculette : محاسب صغير)

مع المعطيات : 20 * 317.961 النتيجة : 635.922
 المعطيات : 1.0 / 0.0 النتيجة : مَأَلَكَةُ الغلط (message d'erreur) ،
 متوقفة على الحاسب المستعمل

7.1 - إختيار طريقة

تسمح الطريقة الإستنتاجية بانطلاقة جيدة في البرمجة وكذلك هي وسيلة نظرية خصبة .

بالنسبة للمبرمج ، تكمن فائدتها في جملة ما ، في أنها تستوجب تفكيراً « على ما نريد صنعه » ، تفكيراً يؤدي الى استنتاجات على « كيف يجب صنعه » . هذا المدخل يُوصَلُ عامة إلى برامج صحيحة إبتداء من التجربة الأولى (هذا ما هو نادراً إذا كنا نبرمج دون إتباع طريقة ما) .

إن التنفيذ اليدوي بالنسبة لبعض أنواع البرامج ، يصبح سريعاً شاقاً للغاية ؛ يجب إذن أن نحفظ المهم من أفكار الطريقة وأن نترك جانباً التحليل التقني . إن الطريقة المقترحة لتكوين برنامجاً تبقى بالفعل فعالة بالكامل إذا تم إهمال التفاصيل الغير متوافقة مع المسألة .

إذا عملنا بهذا الشكل فإننا سنقترب من طرق تحليل أخرى . الأكثر شمولية والأكثر شهرة هي فيما يظهر البرمجة بطريقة التدقيق المتتالي (raffinements successifs) التي سوف نتناولها تباعاً في سياق النص .

8.1 - تمارين

- 1 - أوجد الثمن المتوجب دفعه عند تسليم كمية من الفيول . يجب الأخذ بعين الاعتبار قيمة الضريبة المتوجبة وحسباً بقيمة 3% إذا تعدى المبلغ الإجمالي 2500 فرنك .
- 2 - حلّ المعادلة من الدرجة الأولى $ax + b = 0$ مع الأخذ بعين الاعتبار الحالة التي تكون فيها a مساوية لصفر .
- 3 - حلّ النظام الخطّي للمعادلتين ذات المجهولين :

$$\begin{cases} ax + by = c \\ dx + ey = f \end{cases}$$

وذلك بتطبيق صيغ كرامر (Cramer) :

$$x = (ce - bf) / (ae - bd), y = (af - cd) / (ae - bd)$$

- 4 - إحسب المال الواجب إرجاعه بواسطة موزّع اوتوماتي محتو على عدد غير محدد من القطع النقدية : 10 فرنك ، 5 فرنك ، 2 فرنك و 1 فرنك ، يجب أن نقلل من عدد القطع الواجب إرجاعها .

5 - إطبغ جدولاً لـ N قوة متتالية للقيمة x .

6 - إطبغ جدولاً لقيم المتسلسلة U حيث :

$$u_i = (i + 1)/(i \star i + i + 1) \text{ pour } 0 \leq i \leq n.$$

$$S2_i = \sum_{n=1} n^2 \quad \text{7 - احسب}$$

8 - احسب عامل القيمة الصحيحة المعطية (factorial)

$$\langle F_n = (F_{n-1}) \star N \rangle$$

9 - احسب في \mathbb{R}^n المسافة بين نقطتين A و B محدّتين بإحداثياتهن

$$d(A, B) = \sqrt{\sum_{i=1}^n (A_i - B_i)^2} \quad A_1, A_2, \dots, A_n \text{ et } B_1, B_2, \dots, B_n.$$

10 - إقرأ نصاً ينتهي بنقطة ، وله على الأكثر 40 سمة ومن ثم اكتبه بالمقلوب على أن ينتهي النص المكوّن حيثنذ بنقطة . فمثلاً «lire un texte.» كمعطية تصبح بالمقلوب «etxet . nu eril.»

11 - اضبط (Normaliser) متسلسلة v من القيم الحقيقية : المراد هو إعادة القيم على الفترة [1..0] بواسطة تطبيق خطّي (application linéaire)

$$u_i = kv_i$$

12 - تسمح صيغة جون ماشين (John Machin) بحساب π (

$$\pi = 4 \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} \left(\frac{4}{5^{2n+1}} - \frac{1}{239^{2n+1}} \right)$$

إنها بالفعل عملية تطوير π المساوية لـ :

$$(\pi = 16 \text{ قوس ظل } \frac{1}{5} - 4 \text{ قوس ظل } \frac{1}{239})$$

الفصل الثاني

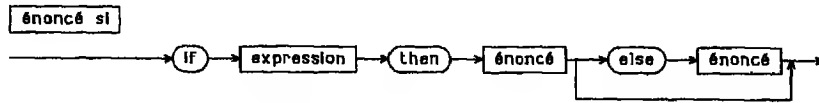
قواعد اللغة

0.2 - كتابة برنامج

عندما نتكلم عن كتابة برنامج ، فإننا نعني حينها (على الأقل) شكلين : النحو والمعنى .

المعنى (Sémantique) ويقصد به كل ما له علاقة بالمدلول ، إنه ما يريد البرنامج قوله ، إنه تسلسل الحسابات التي تؤدي إلى النتيجة . لا تصبح النتيجة سليمة إلا إذا كان تحليل المسألة (التي يحلّها البرنامج) قد تم بشكل صحيح ، إمّا بطريقة صوريّة ، مثل الطريقة الإستنتاجية المشروحة سابقاً ، إمّا بالمهارة التي هي ثمرة التجربة . لا تنطبق هذه الحالة الأخيرة إلا على المسائل السهلة التي يكون قد حلّ مثلها سابقاً .

النحو (Syntaxe) ويقصد به كل ما له علاقة بوظيفة وتنظيم الرموز ، إنه الطريقة التي يجب أن يُكتب بها البرنامج لكي يصبح مفهوماً من قبل الآلة . يتم تحديد نحو البرنامج بواسطة قواعد اللغة المرسومة هنا بواسطة مخططات :



(expression = تعبير ؛ énoncé : عبارة ؛ si : إذا)

فإذا « $\text{if } a = b \text{ then } c := a / 2$ » هي من الناحية النحوية سليمة ، بينما « $\text{if } a =$ » هي ليست كذلك وسيتم الإعلان عنها من قبل الآلة كجملة غير مفهومة .

ننوّط في المخطّط داخل مستطيل ما هو معرّف في مكان آخر ، ونُحيطُ بدائرة الرموز الأساسية للغة (الكلمات الدليليّة «Key-word» والرموز الخاصة مثل : = أو ؛) .

معلومات سابقة

يتناول هذا الفصل ما هو ضروري ولا غنى عنه لكتابة برنامج في الباسكال . إنه لا يستلزم معلومات أخرى عن التحليل غير التي تناولها الفصل الأول أو على الأقل ممارسة للبرمجة في لغة أخرى كالـ BASIC أو الـ FORTRAN مثلاً .

سننطلق من التركيبة العامة للبرنامج ، وسنوضح تباعاً العناصر ، دون أن نسعى للإحاطة بكل اللغة (هذا ما سيتم في الفصل التالي) ولا إلى التفكير بالتحليل أو الخوارزميات أو مجّمع المعطيات .
إنه فصل يعتمد على الوصف ، حيث تتطابق فيه الأمثلة مع تشكيلات أو مع تماثلات بارزة أكثر مما هي تنطبق على برامج كاملة .

1.2 - مفاهيم مبدئية

مثال : برميل

إن المسألة « إحسب حجم برميل » تؤدي الى المواصفة « إحسب حجم البرميل ذي الأبعاد التالية : الطول L ، القطر الكبير D ، القطر الصغير d على أن يحسب حجمه تبعاً للصفة التالية :

$$V = \pi L \left[\frac{d}{2} + \frac{2}{3} \left(\frac{D}{2} - \frac{d}{2} \right) \right]^2$$

لتكوين التحليل ، ننطلق من تعريف النتيجة (أكتب V) ، لحساب V ، نستعمل L ، d و D الذين هم معطيات :

معجم	تعريفات
②	V (حقيقي) : حجم
④	pd (حقيقي) : القطر الصغير gd (حقيقي) : القطر الكبير L (حقيقي) : الطول $3.14159 = pi$ (حقيقي)
⑥	⑤ L, gd, pd في معطية
③	① نتيجة = أكتب V ③ $V = pi * L * \text{sqr} (pd/2 + 2/3 * (gd / 2 - pd / 2))$

في هذا التحليل قمنا بتخطية (Linéarisation) الصيغة الرياضية والتحديد بأن pi هي ثابتة : إننا نستبق على نحو الباسكال . كذلك فإننا أشرنا بـ pd إلى d و gd إلى D ذلك لأن d و D سيكونان كتابتان مختلفتان لنفس المعرف .

إن النقل إلى لغة الباسكال يؤدي إلى تمييز التصريحات ، المتوافقة مع المعجم ، التي تحدد خواصّ المعرفين (π ثابتة ، D, d و L معرفين لمتغيرات ذات قيم في مجموعة الأعداد الحقيقية) والعبارات ، المتوافقة مع التعريفات ، التي تحدد الحسابات الواجب إجراؤها على المتغيرات والثوابت :

البرنامج

```
program tonneau(input,output);
{ حساب حجم البرميل }
const pi=3.14159;
var V:real;      { حجم }
    pd:real;     { القطر الصغير }
    gd:real;     { القطر الكبير }
    L:real;      { الطول }
begin
  read(pd,gd,L);
  V:=pi*L*sqr(pd/2.0+2.0/3.0*(gd/2.0-pd/2.0));
  writeln (V)
end.
```

(tonneau : برميل)

تصريحات وعبارات (Déclarations et énoncés)

يتألف البرنامج من تصريحات تحدد الأدوات المعالجة من قبل العبارات ، وذلك بتحديد طرق استعمالها (ثابت ، متغير ، ...) نوعيتها (عدد صحيح ، عدد حقيقي ، ...) والمعرف المبرر عنها (\dots, V, π) ؛

وعبارات تحدد الأفعال المبدئية الواجب إجراؤها (قراءة ، مقارنة ، كتابة ، تعيين ...) وسلسلتها : متتالية ، شرطية ، ...

ملاحظة : يطلق إسم « تعليمية » على العبارة . إننا نتبع النظم AFNOR التي تستعمل « العبارة » بشكل عام ، يجب التصريح عن كل معرف قبل استعماله .

بعض المصطلحات (un peu de Vocabulaire)

الثابت هو قيمة غير قابلة للتغيير من قبل البرنامج . يجب تمييز الثوابت الصحيحة (4 ، -3 ، 0) ، الثوابت الحقيقية (3.14159) ، الثوابت المسلسلة ('volume = ' هو سلسال من سبع سمات) . يمكن تمثيل الثابت بمعرف لثابت (π) ، محدد بواسطة تصريح لثابت (مُدخل بالكلمة الدليل const) .

في مقابل الثابت ، فإن قيمة المتغير قابلة للتغيير من قبل البرنامج .
المعرف هو الإسم المطلق على أداة ؛ يوجد معرفين لثابت (π) ، لمتغير (v, pd, gd)

لإجراء (writeln, read) . . . يتم حجز بعض المعرفين لغايات خاصة ، إنها الكلمات الدليّة (... , Const, program) .

إن مجموعة القيم التي يمكن أن تأخذها أداة هي نوعها . يمكن للنوع أن يحدد من قبل البرنامج ، أو أن يكن محدّد سلفاً (integer, real : حقيقي ، صحيح) .

إن عبارة التعيين ($V := \pi * I * \dots$) تعيد تعريف القيمة الملحقّة بالمتغيّر المعيّن (V) نتيجة التقييم لتعبير ($\pi * I * \dots$) .

التمثيل الحرفي (représentation typographique)

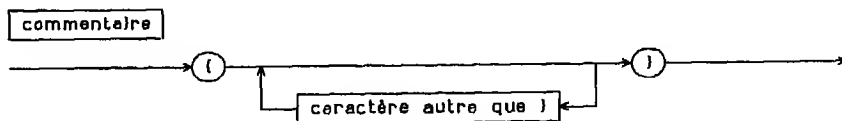
إن كل سمة ظاهرة خارج إطار ثابتة مسلسلة ('volume' أو 'H') لا تجد معناها قد تغيّر بفعل تغيير في التمثيل الحرفي . بوجه خاص ، يمكن بالنسبة لمعرف تمثيل حرف بالشكل الصغير (minuscule) أو بالشكل الكبير (majuscule) : فمثلاً longueur donde و longueur dOnde و LONGUEURdONDE هم ثلاثة كتابات لنفس المعرف ؛ كذلك 14 ، 14 ، 14 و 14 هم نفس الثابت .

تبعاً للأدوات المستعملة ، يمكن توفير طريقة أو عدة طرق تلاعب بالسّمات الحرفية (غليظ صغير ، كبير ، . . .) ؛ عامة فإن الأحرف الكبيرة هي متوفرة دوماً .

كل تلك الطرق تهدف الى تأمين قراءة جيدة للبرنامج وهذا ما نستعمله في كتابنا هذا عندما نكتب البرامج إلّا في حالة البرامج المختبرة الكاملة حيث أن الطابعة المستعملة لا تسمح بتلاعب في التمثيلات الحرفيّة .

ملاحظات (Commentaires)

تستعمل الملاحظة بهدف تسهيل فهم البرنامج من قبل القارئ العادي ، بينما تبقى دون أي تأثير فيما يخصّ مدلول البرنامج بالنسبة للحاسب الآلي .



(commentaire : ملاحظة ؛ caractère autre que : سمة غير)

إن الملاحظات ، في الحالة الراهنة للتكنولوجيا ، هنّ غير مفهومات من قبل الحاسب الآلي (هذا ما يدعو إلى إهمالهن) ، بينما هن ربما الجزء الأساسي في البرنامج .

يمكن أن يتم إدخال الملاحظات في أي مكان حيث يمكن ، أو يجب ، وضع تباعد (سمة فراغ ' ') .

تباعدات (Espaces)

إن الملاحظات ، التباعدات (ما عدا تلك التي توجد في سلسلات السمات) ونهايات السطور هن فواصل لوحداث من مفردات اللغة . يمكن أن تظهر تلك الفواصل :
- مرة واحدة على الأقل بين وحدتين متتاليتين واللتين تكونان معرفين ، كلمات دلالية ، وسومات أو أعداد دون علامة ؛
- مرة واحدة على الأكثر بين وحدتين من مفردات اللغة .

يمنع وضعهم داخل وحدات مفردات اللغة ؛ فمثلاً لا يمكن إدخال فراغ ، ملاحظة أو نهاية سطر في معرف ما ، يجب أن نكتب « : = » دون أي تباعد .
غالباً ما نوجز هذه القواعد بالقول « ان » لغة الباسكال هي لغة ذات نسق حر » .
ملاحظة : نجد في الملحق 3 قائمة الرموز الخاصة والكلمات الدلالية .

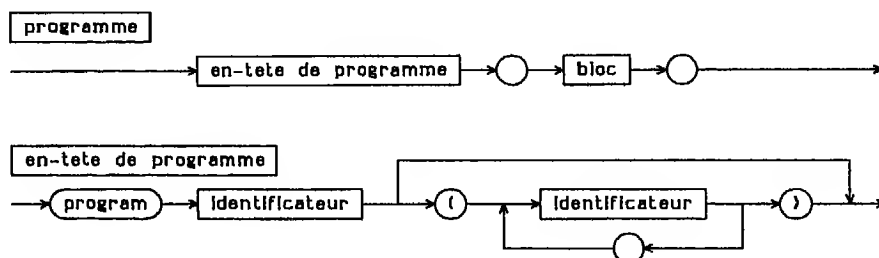
2.2 - التكوين الإجمالي للبرنامج

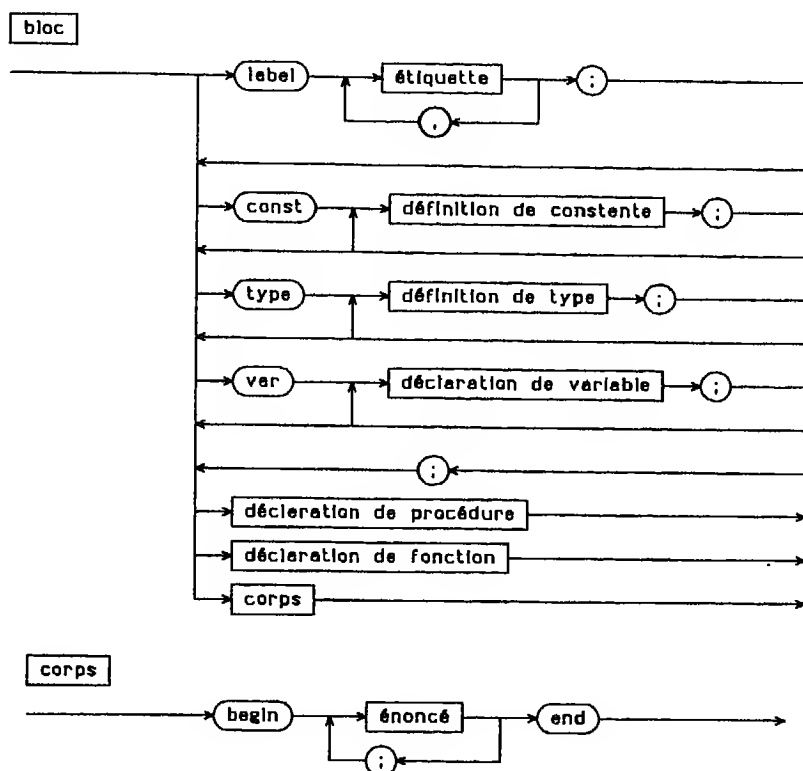
البرنامج (Programme)

تظهر مختلف أجزاء البرنامج ضمن ترتيب محدد مفروض من قبل قواعد النحو :

- 1 - عنوان البرنامج
 - 2 أقسام التصريحات .
 - 3 - قسم التعابير .
- يطلق إسم جسم على قسم التعابير ، وفدرة على أقسام التصريحات والتعابير .
تتألف أقسام التصريحات من :

- 1 - قسم تصريح الوسومات (أنظر 3.3.3)
- 2 - قسم تعريفات الثوابت (أنظر 1.3.2)
- 3 - قسم تعريفات الأنواع (أنظر 4.3.2)
- 4 - قسم تصريحات المتغيرات (أنظر 2.3.2) .
- 5 - قسم تصريحات الإجراءات والدوال (أنظر 4.3) .





(programme : برنامج ؛ en-tête : عنوان ؛ bloc : فقرة ؛ معرف : identificateur ؛ معرف : étiquette ؛ label ؛
 وسم : déclaration ؛ تصريح : définition تعريف ؛ type : نوع ؛ variable : متغير ؛ procédure :
 إجراء ؛ fonction : دالة ؛ corps : جسم ؛ énoncé : تعبير) .

إذا تفحصنا مخططات النحو هذه ، يمكن التحقق من كتابة البرنامج التالي :

البرنامج

```

program S2(input,output);
{ مجموع عددين }
var a,b,S:real;
begin read(a,b); S:=a+b; writeln(S) end.

```

- تم إحترام ترتيب مختلف الأقسام .
- الأقسام المدخلة من قبل الكلمات الدليلية label ، const ، type هي غير إلزامية .
- في نهاية البرنامج ، في «end.» ، تدخل «end» ضمن إطار الجسم بينما « . » لا تدخل .

سيتم غالباً في ما بعد إعادة إستعمال هذه المخططات ؛ بالأخص فإن تصريح الإجراء أو الدالة يتضمن فدره .
 فيما يخص التحليل ، فإن الجسم يتوافق مع تعريفات التحليل ، وأقسام التصريحات تتوافق مع المعجم . أما الإجراءات والدوال فسيتم إخراجها عند الضرورة من جداول ثانوية .

عنوان البرنامج (L'en-tête du programme)

يحدد عنوان البرنامج الإسم المعطى للبرنامج ، إضافة إلى الأدوات الخارجية عن البرنامج التي سيتم استعمالها .
 تسمح قائمة المعرفين بين المزدوجات بالعمل على التطابق ما بين أدوات داخلية في البرنامج (مسندة بواسطة معرفيها) وأدوات خارجية . فمثلاً

program néant (input, output);

يجري إتصلاً بين أعضاء الإدخال والإخراج التابعة للحاسوب ، وبين السجلات ذات الإسم input و output المستعملة ضمناً في البرنامج في تعابير القراءة (read) والكتابة (writeln) .

إن المعرف néant الذي يلي الكلمة الدليلية Program ليس له مدلول في داخل البرنامج .

الفدره (Le bloc)

إنها تصف أدوات (وسم ، ثابت ، نوع ، متغير) وأفعال (تعابير) . إنها تحمل إسماً محدداً بالرأس السابق للبرنامج إذا كان المقصود فدره من البرنامج ، أو بعنوان الإجراء (أو الدالة) إذا كان المقصود فدره من الإجراء (أو الدالة) .

إجراء أو دالة ، دون وسيط (Procédure ou fonction, sans paramètre)

إن عملية التصريح عن الإجراء أو الدالة تسمح بإعطاء إسم إلى جزء من البرنامج . بذلك فإنه من غير الضروري في ما يلي ، إعادة كتابة هذه التعابير عند كل إستعمال ، بل نكتفي بنداء الإجراء (أو الدالة) عن طريق تسميته :

```
... read(a,b); writeln('a=',a,' b=',b);
repeat ...
... until ...
read(a,b); writeln('a=',a,' b=',b); ...
```

يمكن كتابة الفقرة السابقة بشكل أسهل كما يلي :

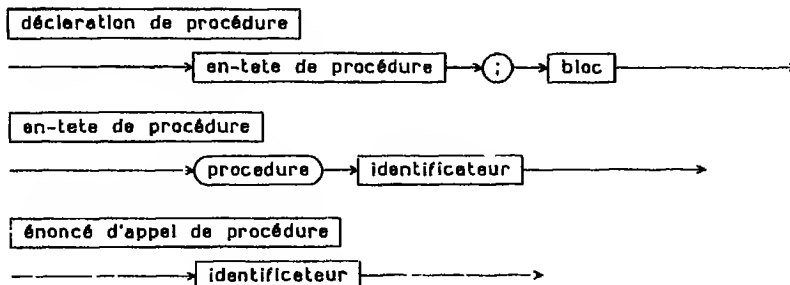
```

procedure ab;
begin read(a,b); writeln('a=',a,' b=',b) end; } تصريح
...
...
ab;
repeat...
... until ...
ab; ...

```

نداء →

نداء →



لقد تم إعطاء نحو الفدرة سابقاً . يمكن للفدرة أن تحتوي على تصريحات للإجراءات . يحتوي تصريح الإجراء على فدرة ، التي بدورها تحتوي على تصريحات لإجراءات ... الخ . تسمى قاعدة النحو الخاص بالفدرة بالتكرارية .

3.2 - الأدوات المعالجة

0.3.2 - أنواع (types)

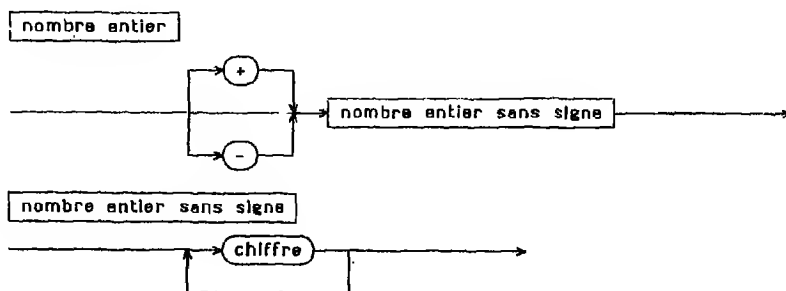
يجري الجسم حساباً على أدوات : ثوابت ومتغيرات . تتحرك قيمة المتغير في إطار محدد بنوع . يوجد أنواع بسيطة ، أخرى محددة مسبقاً (صحيح ، حقيقي ، سمة ، بولي) ، البعض الآخر منشأ من قبل المبرمج (فترة ، عدّ) ؛ يوجد أيضاً أنواع مركبة ، منشأة من قبل المبرمج إنطلاقاً من أنواع بسيطة ، مثل الجداول ، أو محددة مسبقاً ، مثل النصوص . بعض التركيبات الأخرى الأقل إستعمالاً ، سيتم بحثها في الفصل 4 : مجموعات ، دلائل وسجلات .

إنّته : لا يمكن ، في تعبير ، في تعيين أو في مقارنة ، معالجة أدوات مجتمعة إلا إذا كانت ذوات أنواع متساوقة ؛ سيتم بحث هذه الفكرة الصعبة بعض الشيء في آخر الفقرة .

1.3.2 - ثوابت (Constantes)

لا يستطيع البرنامج تغيير أي ثابت ؛ يمكن أن يكون الثابت عدداً ، سمة ، سلسلاً أو معرفاً لثابت .

يُخضع العدد الصحيح ، عنصر من النوع integer ، إلى النحو :



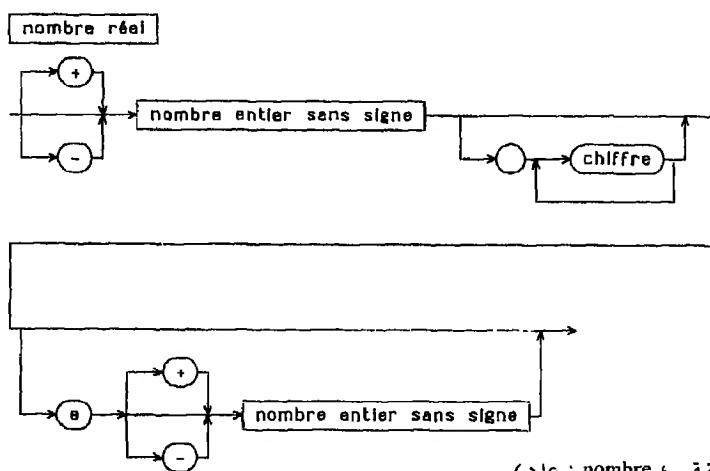
(chiffre : عدد ؛ signe : علامة ؛ entier : صحيح)

وذلك ضمن التمثيل في القاعدة 10 .

مثال : 0 - 3 4

لكن هناك نهاية ، معروفة بالثابت المحدد مسبقاً maxint ، لقيمة العدد الصحيح ؛
إن عدداً صحيحاً دون علامة ينتمي إلى الفترة 0... maxint .

يتوافق العدد الحقيقي ، عنصر من النوع real ، مع قيمة غير صحيحة ، ويمكن كتابته بشكل ثابت ، مثل 3.14 ، أو بشكل طليق الفاصلة ، مثل 0.314E1 حيث تعني «E» مضروباً بـ 10 مرفوعاً إلى قوة : 0.314×10^1 (النقطة الأنكلوسكسونية تحل مكان الفاصلة الفرنسية) .



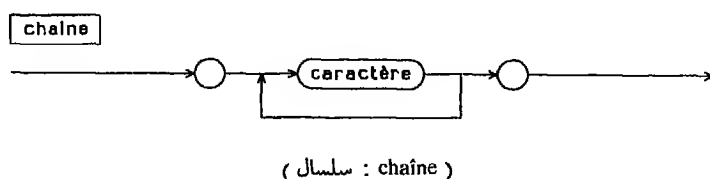
(réel : حقيقي ؛ nombre : عدد)

مثال : 3.14159 - 17 27.9E - 5E-2 0.1E7 0.1e7 1.0

إنتبه : 1. و 1. هم كتابات غير صحيحة

1.0 0.001E3 و 10.0000E - 1 تعني نفس العدد

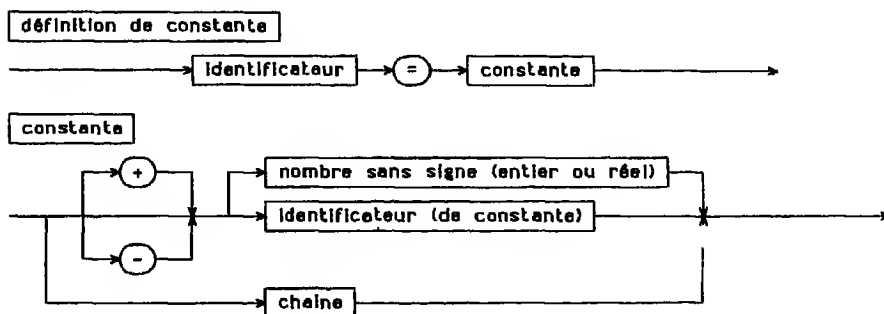
إن سلسلاً من n سمات هو متسلسلة من n سمات ، مأخوذة ضمن إطار لعب خاص بكل حاسب آلي . إنه العنصر الوحيد في اللغة (مع الملاحظات) الذي يلعب فيه التمثيل الحرفي دوراً مهماً . ننوّط السلسال عن طريق إحاطته بعلامات حذف (التي لا تدخل ضمن نطاق السلسال) ومن المتعارف عليه أن يتم تكرير علامة الحذف عند وجودها في سلسال : مثلاً السلسال aujourd'hui الشكل التالي 'aujourd'hui' .



إن سلسلاً من سمة واحدة هو قيمة من النوع المحدد مسبقاً char (سمة) .

تعريف الثابت

إنه عملية تصريح ، مُدخلة بواسطة الكلمة الدلالية const التي تربط معرفاً بقيمة . فيما يلي ، يكون المعرف الذي تم تعريفه ، كناية عن تنويط آخر للثابت .



(définition de constante : تعريف الثابت identificateur : معرف) .

مثال :

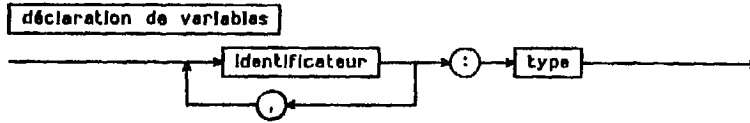
؛ ' ' blanc (فراغ) N = 3 ؛ e = 2.71828 const

بالعكس N1.004 N + 1 هي كتابة غير مشرّعة .

2.3.2 - متغيرات

بعكس الثابت ، فإن المتغير هو كناية عن كيان تعيّن له قيمة ، يجب التصريح عن كل متغير ، وذلك بتحديد المعرف الذي يُسمّيه ، ونوع القيمة الممكن تعيينها .

تم عملية تصريح المتغيرات عن طريق إدخال الكلمة الدلالية `var` :



مثال :

`Var i : integer ;` (المعرف للمتغير `i` هو من النوع `integer`) (صحيح)

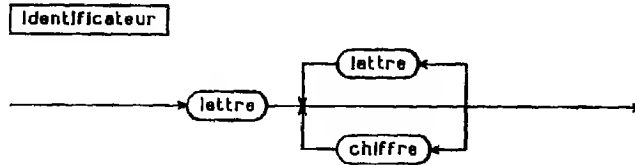
`x, y, z : real ;` (المعرفين `z, y, x` هم من النوع `real` (حقيقي)

`a, b : T` هي كتابة مختصرة لـ `a : T ; b : T ;`

لا يمكن إستخدام قيمة متغيرٍ إلا إذا تم تحديدها : إن العملية الأولى التي تتم على متغير ما في برنامج ما هي بالضرورة عملية تعيين (أو عملية مشابهة كالقراءة `read`) .

المعرف

إن جميع السمات الخاصة بمعرف هي ذات مدلول :



(`identificateur` : معرف ؛ `lettre` : حرف ؛ `chiffre` : رقم)

لا يمكن لأي معرفٍ أن تكون لديه نفس الكتابة الخاصة بكلمة دلالية (أنظر الملحق

3) ؛ بعض المعرفين المسمّين `Prédéfinis` (محددين مسبقاً) (مثل `read` ، `maxint` ، `integer`) لهم مسبقاً مدلولاً خاصاً ، هذا لا يمنع من إمكانية إعادة تحديدهم .

مثال : `Chapitre3 X3 heure`

بالعكس `var un_peu d'accord Chapitre3.2 3X`

ليسوا معرّفين أصحاء .
ملاحظة : يوجد بعض النظم التي تحدد عدد السمات ذات المدلول بـ 10 (CDC) أو بـ 8 (USCD)

المظاهر الساكنة والمتحركة

إن التعيين $u : u + 1$ ، حيث تمت زيادة قيمة المتغير u ، يعرض نظرة تحريكية (dynamique) : الأداة u المعيّنة ليست نفسها أداة الإنطلاق ، إنها تأتي بعدها في تسلسل الوقت . بالفعل فإن المقصود هو عملية حسابية على متسلسلة $u_i = u_{i-1} + 1$ حيث أن الدليل i يكون رقم ترتيب التغيرات المجراة على المتغير u ؛ هذا نما نَوَّطَه بـ $u = u + 1$ بالطريقة الساكنة (statique) في التحليل الإستنتاجي . هذا المظهر المتحرك هو أساسي في عملية فهم فكرة المتغير ؛ من الأفضل التفكير بطريقة سكونية بدل أن نعالج مباشرة أدوات متحركة .

3.3.2 - أنواع بسيطة

يُحدّد النوع البسيط مجموعة منظّمة من القيم . بعض الأنواع ، المسماة أوليّة ، هي محددة مسبقاً (كانت موجودة سابقاً في البرنامج) : صحيح ، حقيقي ، بولي وسمّة ، بعضها الآخر يتم تحديده بواسطة البرنامج الذي يستعملها : أنواع فترات وتعداد .

النوع الحقيقي : real

تؤلف القيم مجموعة ثانوية ، محددة تبعاً لكل حاسب آلي ، من مجموعة الأعداد الحقيقية

ملاحظة : تبعاً للحاسب المستعمل ، تتنوع دقة الأعداد الحقيقية ؛ فمثلاً بتمثيل « على 32 بّته » ، يجب أن تكون القيمة المطلقة للأسّ أقل من 37 ، وأن يكون للجزء العشري ستة أرقام فقط (في القاعدة 10) ذات ملالول :

$$(\text{« على 32 بّته »}) \quad 10^{37} < r < 10^{-37} \text{ أو } -10^{-37} < r < 10^{37}$$

فمثلاً 1.000 000 و 1.0000 0001 هما بالنسبة للحاسب نفس الرقم . أما المؤثرات المطبقة على النوع الصحيح فهي + ، - ، * ، / .

النوع صحيح : Integer (نوع ترتيبي (ordinal))

تكون القيم مجموعة ثانوية من الأعداد الصحيحة ، محدّدة بالفترة .. maxint. ، حيث أن maxint هي ثابتة محدّدة مسبقاً ومتعلقة بنوعية الحاسب المستعمل .

ملاحظة : على حاسب « ذو كلمات من 16 بّته » تكون $\text{maxint} = 32767$ ، بينما إذا كان « ذو كلمات من 32 بّته » فإن $\text{maxint} = 107321823$.

إن المؤثرات + ، - ، * ، div mod تطبّق على النوع صحيح . إن النوع صحيح هونوع ترتيبي ، مثل الأنواع بولي ، سمة ، تعداد (وفترات) : يمكن أن نطبّق على نوع ترتيبي ، الدوال succ, pred و ord التي تعطي تباعاً السلف ، الخلف والعدد الترتيبي المشترك بقيمة .

النوع بولي : booléan (نوع ترتيبي)

إن القيم هي false (خطأ) و true (صح) ، إنهم معرفين لثوابت محدّدة مسبقاً حيث false تسبق true .

الخصائص :
 $ord(false) = 0$ $ord(true) = 1$
 $succ(false) = true$ $pred(true) = false$
 $false < true$

تُطبّق العمليات المنطقية العادية على النوع بولي : and (و) ، or (و) ، not (لا) ملاحظة : تعني false باللغة الفرنسية «Faux» و true تعني «Vrai» .

النوع سمة : char (نوع ترتيبي)

إنه تعداد للعب السمات (خاص بكل آلة) ، التي يكون للبعض منها تمثيل تخطيطي (أحرف ، أرقام ، ...) ولا للبعض الآخر (سمات « ضبط » النقل : RETURN ، ESCAPE ، ...) .

تُحدّد الأعداد الترتيبية للسمات بالتكويد المستعمل (ASCII ، EBCDIC ، BCD ... انظر الملحق 1) ؛ إنها موجبة ، تبدأ من الصفر ، وهي متتالية .

الكتابة : بالنسبة للسمات « القابلة للطبع » : نضعها بين علامات حذف مثال : 'A' '2' '—'

فيما خص السمات « غير القابلة للطبع » : لا يوجد تنويط محسوب (لكن الدالة chr تسمح بتخطي الصعوبة) .

إنه نوع ترتيبي ، فإذا تطبق عليه الدوال succ, pred و ord ؛ نشير إلى أن لدى ord دالة مُعاكسة ، تنوّط chr : $c = chr(ord(c))$.

بالنسبة لترتيب السمات ، فإننا لا نعرف إلا الخصائص التالية :

1 - تكون الأرقام منظمة ومتلاصقة :

$$succ('0') = '1' \dots '0' < '1' < \dots < '9'$$

2 - تكون الأحرف الكبيرة ، إذا كانت موجودة في الحاسب الآلي ، منظمة ، لكن ليس بالضرورة متلاصقة :

$$'A' < 'B' < \dots < 'Z'$$

3 - نفس الخاصة بالنسبة للأحرف الصغيرة في حال وجودها في الحاسب .

$$'a' < 'b' < \dots < 'z'$$

4 - تعطي مقارنة سمتين نفس نتيجة مقارنة أعدادهن الترتيبية .

$$c < d \Leftrightarrow \text{ord}(c) < \text{ord}(d)$$

إننا لا نعرف إذن أي شيء ذي صفة عامة حول ترتيب أي سمتين (العلاقة ما بين الأرقام والأحرف ، ما بين تمثيلين مختلفين للأحرف ، موضع التباعد (' ') في اللعبة ، . .) : تختلف هذه العلاقات من حاسب إلى آخر .

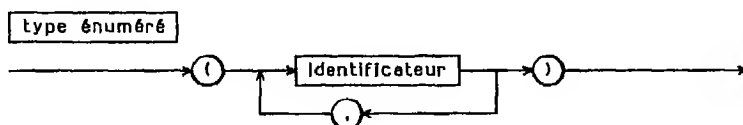
الأنواع تعداد (نوع ترتيبي) (Types énumérés) إنه مجموعة منظمة من القيم ، المحددة بالتعداد للمعرفين الذين يعبروا عن هذه القيم . مثال :

var Jour : (lundi, mardi, mercredi , jeudi, vendredi)

الجمعة ، الخميس ، الأربعاء ، الثلاثاء ، الإثنين ، نهار

يمكن أن يأخذ المتغير Jour فقط إحدى القيم من Lundi حتى vendredi ويمكننا كتابة $\text{succ}(\text{jour}) = \text{jour}$ ، مع الإشارة بأن lundi ليس له سلف وvendredi ليس له خلف .

تكون الأعداد الترتيبية متتالية وتبدأ من الصفر :



(type énuméré : نوع تعداد ؛ identificateur : معرف)

كما يظهر فإن هذه الأنواع تعداد فائدة كبيرة : إنها تغني عن توكيدات مملّة التي هي بالأغلب مصادر للأخطاء .

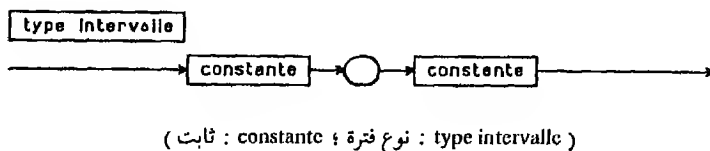
الأنواع فترات (ترتيبية) (type intervalles)

إنه يحدّد فترة من القيم المتتالية ، المأخوذة في نوع ترتيبي ، يسمى نوع سائد (hôte) ؛ تنتمي الحدود الدنيا والقصوى المستعملة إلى الفترة التي ترث العمليات المطبّقة على النوع سائد (host) .

مثال :

$$1 \dots 100 \quad -10 \dots +10 \quad '0' \dots '9'$$

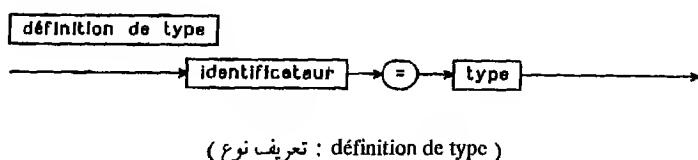
لا يمكن لمتغير مصرح من داخل نوع فترة أن يأخذ قيمة خارج هذه الفترة . سنجد هذه الخاصة عند دلائل الجداول .



مثال : مع معرفة النوع (lundi, mardi, mercredi, jeudi, vendredi) ، يمكن تكوين الأنواع lundi ... jeudi أو mardi ... jeudi .

4.3.2 - تعريف نوع

يسمح تعريف نوع بربط اسم (معرف) بنوع . تدخل هذه التعريفات بواسطة الكلمة الدلالية type .



هكذا يمكن إستعمال الإسم المعطى إلى النوع في كل مرة نحتاج فيها إلى وصف النوع : لإنشاء نوع مركب ، في تصريح متغيرات ، ...
مثال :

```
type reponse=(oui,non,peutEtre);
chiffre='0'..'9';
typeSimple=(reel,entier,booleen,caractere);
typeOrdinal=entree..caractere;
```

(réponse : جواب ; chiffre : رقم ; simple : بسيط ; ordinal : ترتيب)

فأذن : var dif: chiffre ، محل مكان '0' ... '9' ;

يكون النوع البولي محدد مسبقاً بشكل منسّق وخارج إطار البرنامج وذلك بواسطة
type boolean = (false, true);

تعريف واحد

كل معرف ، بعد أن تم تصريجه أو تعريفه ، لا يمكن أن يُعد نفسه في نفس النص .
هكذا ومع التعريفات السابقة ، فإنه من غير الممكن كتابة :
var réel : real; (سيعني ذلك إعادة تعريف (réel))

أو ; `var scalaire: (booléen , énuméré)` (سيعني ذلك إعادة تعريف
 . `(booléen`

ملاحظة : هذه القاعدة لا تطبق بتاتاً بمجرد أن تغيرت القدرة .

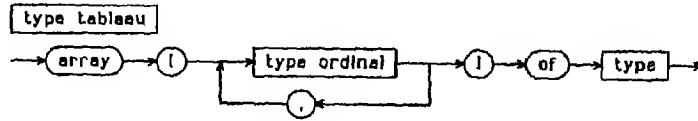
5.3.2 - تكوين الجدول

الجدول كناية عن مجموعة من المركبات جميعها من نفس النوع ، ولها عدد ثابت معروف مسبقاً . يرتبط عدد المركبات بفترة تغيير الدليل .

التصريح : نوع المركبات `of` [نوع الدليل] `array`

- لا يمكن أن يكون نوع الدليل صحيحاً (لكن يمكن أن يكون فترة) ولا حقيقياً ؛ يجب أن يكون لديه نوعاً بسيطاً ،

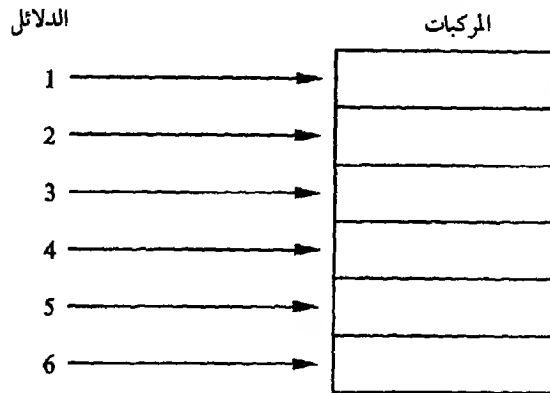
- إن نوع المركبات هو غير محدد (أي كان) ، يمكن أن يكون جدولاً .



(tableau : جدول ؛ ordinal : ترتيب) .

إحداثيات نقطة في R^6 : `array [1..6] of real`

توافق فكرة الجدول مع تطبيق لكل قيمة من نوع الدليل على مركب مختلف :



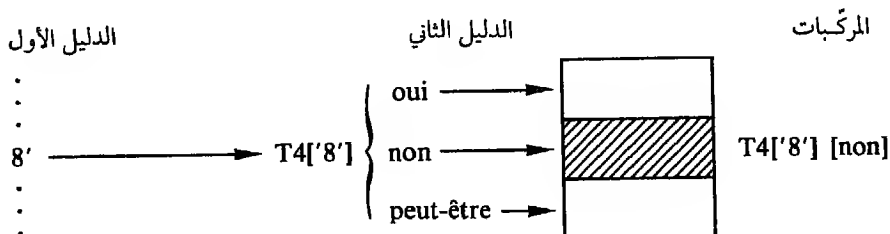
نستعمل الجدول عندما نريد تخزين قيم ذوات عدد محدد وكلها من نفس النوع :
 علامات إمتحان ، n كشف للحرارة ، ...
 يمكن أن يكون نوع الدليل

- فترة
 array [0..99] of integer
 array ['A'..'Z'] of real
 الكترون
 بروتون
 نوترون
 array [(neutron,proton,electron)] of...
 - تعداد
 - نوع معرف سابقاً
 type reponse=(oui,non,peutEtre);
 chiffre='0'..'9';
 T1=array [reponse] of integer;
 var T2: array [chiffre] of boolean;
 T3: array [boolean] of real;
 T4: array [char] of T1;
 (oui : نعم ؛ non : كلا ؛ peut Etre : ربما)

نستعمل مركباً لتغير من نوع جدول عن طريق تدوين دليله بين معقفين
 مثال : آخذين بعين الاعتبار التعريفات السابقة ، يمكن كتابة :

T2 ['5']:=true;
 T3 [true]:=0.0; T3 [T2 ['5']] :=0.0;
 T4 ['8'] [non] :=109;

T4 هو جدول من جدول (نسميه جدولاً ذي بُعدين) :



T4 هو من النوع array [char] of array [réponse] of integer

T4 ['8'] من النوع array [réponse] of integer

T4 ['8'] [non] من النوع integer

إختصارات :

1- في حال كان مركب الجدول جدولاً ، يمكننا عند كتابة المتغير الدليلي ، استبدال

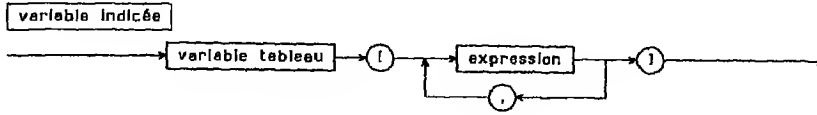
[] بـ -

مثال : T4 ['8', non] هو مكافئ لـ T4 ['8'] [non]

2 - يمكننا ، في خلال عملية تعريف النوع ، إختصار [array] إلى ،

مثال : array [char] of array [reponse] of integer

هو مكافئ لـ array [char, reponse] of integer



(variable indicée : متغير دليلي ؛ tableau : جدول ؛ expression : تعبير)

أمثلة : مصفوفة 10×10 :

M : array [1..10, 1.. 10] of real;

- كمية المعادن الموجودة في الجزر من خلال 20 عملية سبر ، مع 30 جزيرة في كل عملية :

array [(Fe, Cu, As, Ag, E), 1...20, 1...30] of real;

- جداول القيم لدالة عُنْدِيَّة من متغيرين صحيحين :

f: array [0..100, 0..10] of

array [(reel, imaginaire)] of real;

(réel : حقيقي ؛ imaginaire : خيالي)

- نتيجة سيرٍ للرأي العام مؤلف من 60 سؤالاً :

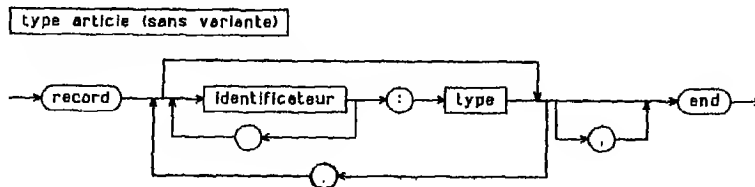
array [1..60] of (oui, non, peut Etre);

(سنرى النوع سلسال في الفقرة 3.4) .

6.3.2 - تكوين الفقرة (article)

تكون الفقرة مجموعة من مركبات ذوات نوع غير محدد (أي كان) ، ولها عدد محدد مسبقاً .

يمكن أن تكون هذه المركبات من أنواع مختلفة ولذا فإنه لا يمكن بلوغها بدليل موحد ، بل بإسم ، يسمى مركب الفقرة بالحقل .



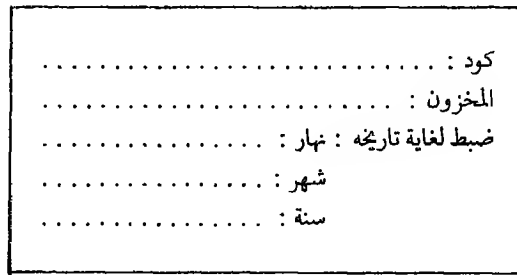
مثال :

```
type date= record
    jour:1..31; mois:1..12; an:1840..1999
end;
modele= record
    code:0..9999; enStock:integer;
    miseAJour:date
end;
var stock: array[1..200] of modele;
```

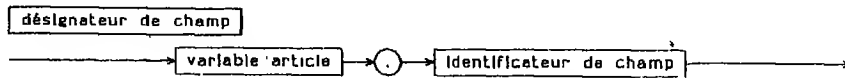
(record : قياس ؛ date : تاريخ ؛ modèle : نموذج ؛ stock : مخزون ؛ jour : نهار ؛ mois : شهر ؛ an : سنة ؛ mise A jour : ضبط لغاية تاريخه ؛ code : كود ؛ variante : مشتق) .

بالحدس ، يمكن التفكير بالفقرة كما لو أنها كانت « سجلاً » من الكرتون :
مخزون : مجموعة من 200 سجلاً من النموذج

نموذج :



يُكتب البلوغ في متغير فقرة على شكل مؤشر للحقل :



(désignateur de champ : مؤشر للحقل ؛ variable article : متغير فقرة)

هو متغير فقرة ؛ فإذاً :	مثال : stock [i]
هو من النوع 0..9999	stock [i] .Code
هو من النوع integer	stock [i] . en stock
هو متغير فقرة	stock [i] . mise A jour
هو من النوع 1.. 31	stock [i] . mise A jour. jour

تُسَعمل الفقرة عندما يجب تخزين قيم من نوع غير محدد وبعدهد معروف .
ملاحظة : سيتم شرح مشتقات الفقرة في 5.4 .

الفقرة والجدول (Article et tableau)

كما رأينا ، فإنه تتشابه كثيراً فكرة الفقرة والجدول ؛ فالمقصود في الحالتين هو مجموعة من المتغيرات ذوات عدد ثابت .

هكذا ، يمكن تمثيل عدد عقديّ بواسطة

جدول : **Var plext: array [(réel, imaginaire)] of real;**

أو فقرة : **Var plexa: record réel, imaginaire: real end;**

هذا ما يؤدي إلى كتابة [réel] plext . réel plexa ؛ يتم الاختيار تبعاً للاستعمال الذي يجريه برنامج الأدوات العقدية .

ملاحظة : فيما خصّ الحاسب ، تمثل الفقرات والجدول بنفس الطريقة داخل الذاكرة ؛ البلوغ يختلف :

- بالنسبة للجدول فإن عنوان $T[x]$ = عنوان أول عنصر من $T + (x - \text{الحد الأدنى}) * \text{حجم المركب}$.

- بالنسبة للفقرة فإن عنوان $A.x$ = عنوان أول عنصر من $A + \text{إنتقال الحقل } x \text{ في الفقرة}$ (قيمة معروفة قبل تنفيذ البرنامج) .

كما نرى فإن بلوغ حقل من الفقرة هو أسرع من بلوغ مكون من الجدول ، خصوصاً عندما يكون جدولاً ذي عدة أبعاد :

var T: array [a..b, c..d, e..f] of ty

alors adresse de T[x, y, z] = adresse T[a, c, e] +

(x - a) ★ (d - c + 1) ★ (f - e + 1) ★ taille d'un composant +

(y - c) ★ (f - e + 1) ★ taille d'un composant +

(z - e) ★ taille d'un composant

(adresse : عنوان ؛ taille d'un composant : حجم المركب)

7.3.2 - قواعد التوافق (compatibilité)

يكون النوعان T1 و T2 متساويين ، إذا كان لهما إحدى هذه الخصائص :

أ - T1 و T2 هما نفس النوع (لهما نفس الاسم)

ب - الواحد هو « فترة » من الآخر ، أو الإثنين معاً هما « فترات » من نفس النوع .

ج - هما نوعا مجموعات منشآت على أنواع أساسية متساوية ؛ T1 و T2 كلاهما معلّين ، أولاً .

د - هما نوعا متسلسلات لهما نفس عدد المركبات .

هكذا وبالعكس الحدس ، فإن نوعين لديهما نفس التكوين ، ليسا بالضرورة متساويين :

type T1 : record x, y : array [0..1] of char end;

Var a : rdcord x, y : array [0..1] of char end;

b : T1

بل يجب أن نكتب :

Var a : T1 ;

b : T1 ;

فيما يلي سنستعمل كثيراً قواعد التساوق هذه خصوصاً قواعد التساوق المتعلقة بالتعيين (1.3) .

4.2 - الدُّخْل - الخُرْج : (OUTPUT-INPUT)

السجل هو كناية عن متسلسلة مركّبات كلها من نفس النوع وبعدها غير محدد (بعكس الجدول) ؛ النص هو كناية عن سجل سمات ، مهيكّل بشكل أسطر .

يتم التعريف المسبق لنصّين في برنامج باسكال :

input ، نص معطيات ، يتم بلوغه بالقراءة بواسطة الإجراء read (إقرأ) ،
output ، نص نتائج ، يتم بلوغه بالكتابة بواسطة الإجراء write (اكتب) . تتوافق هذه النصوص مع :

إستعمال متحرك	إستعمال جامد	
ملامس	قارئ البطاقات	الدخل
الشاشة	الطابع	الخروج

تكون النصوص input و output ، في البرنامج وفي حال استعمالهما ، أدوات خارجية ويجب أن تظهر كوسيط للتصريح **program** ؛ وبذلك فإن علاقة ستنشأ عند التنفيذ بين الأداة الخارجية والأداة الداخلية والتي ستعالج بواسطة الإجراءات المعروفة سابقاً **read** و **write** (وكذلك **readln** ، **writeln** ، ...) .

النص input هو بالشأن معاينة **inspection** : لا يمكن إستعماله إلا للقراءة .
النص output هو بالشأن نتائج **génération** : لا يمكن استعماله إلا للكتابة مع العلم بأنه فارغ في البدء .

1.4.2 - نهاية السجل

بكل نوع سجل (نص مثلاً) نُشَرِك في كل لحظة :

- متسلسلة من القيم
- موقع في المتسلسلة ؛ هذا الموقع يتطابق مع موضع النافذة
- شأن ، معاينة أو نتائج .

تسمح أوليات النيل في السجل بإزاحة النافذة ضمن متسلسلة القيم ، بقيمة واحدة في كل مرة .

في الشأن معاينة («input» مثلاً) ، لا يكون النيل ممكناً إلا إذا لم تصل النافذة إلى نهاية السجل ، أي أنه ما زال يوجد مركبات مطلوب قراءتها .
في الشأن نتائج («output» مثلاً) ، لا يكون النيل ممكناً إلا إذا كانت النافذة في نهاية السجل ، أي أنه من الممكن توسيع السجل .

تسمح الدالة المعرفة مسبقاً eof ، ذات النتيجة البولية ، بمعرفة قيمة هذا الشرط « نهاية السجل » (End Of File بالإنكليزية) ؛ عند كتابتها بدون وسيط ، تُطبّق eof على النص input .

$$\text{eof}(f) = \text{true} \Leftrightarrow \begin{cases} \text{كتابة (write) ممكّنة على السجل } f \\ \text{والشأن نتائج} \end{cases}$$

$$\text{eof}(f) = \text{false} \Leftrightarrow \begin{cases} \text{قراءة (read) ممكّنة على السجل } f \\ \text{والشأن معاينة} \end{cases}$$

مثال : قراءه لكل السمات الموجودة في السجل input ، مع متغيّر C من النوع char (سمة) :

```
While not eof do begin
  read (C) ; { إستعمال السمة المقروءة . . . } end
```

في البداية ، يكون لدينا «eof (output)» و «not eof (input)» (شرط أن لا يكون النص input فارغاً) .

2.4.2 - نهاية السطر (Fin de ligne)

النص هو كناية عن سجل من السمات ، مهيكّل بشكل أسطر .
في الشأن نتائج ، نعرّف نهاية السطر عن طريق إستعمال الإجراء المعرّف مسبقاً writeln ؛ إنها الطريقة الوحيدة (مع page وrewrite) لتعريف نهاية السطر .
في الشأن معاينة ، تسمح الدالة المعرفة مسبقاً eoln ، وعند كل موضع للنافذة ،

بمعرفة قيمة الشرط نهاية السطر (End Of Line بالإنكليزية) في حال أعطت القيمة `coln` القيمة `true` ، فإن السمة المقروءة هي تباعد (espace) . هذا المركب الخاص « نهاية السطر » (مقروء كما لو أنه تباعد) لا يمكن تفرقه عن المركب العادي « تباعد » إلا إذا استعملنا الدالة `coln` .

مثال : مع المتغير `c` من النوع `char` والمحتوى التالي للسجل `input` :

```
PETITbAbPETIT,
bL'OISEAUbFAITb
bSONbNID.
```

(حيث أننا رمزنا إلى السمة تباعد بالحرف `b`)
`while not eof do begin read (c); write (c) end - 1`
 تعطي النتيجة

`PETITbAbPETIT,bbL'OISEAUbFAITbbbSONbNID.b`

```
2) 1. while not eof do begin
    2.   while not coln do begin
    3.     read(c); write(c) end;
    4.   read(c); write('^!^') end
```

تعطي النتيجة

`PETITbAbPETIT,!bL'OISEAUbFAITb!bSONbNID.!`

(تصبح `coln` صح «true» للمرة الأولى عندما تعطي `read (c)` ، في السطر 3 ، السمة '،' `c =` ؛ بينما تسمح `read (c)` في السطر 4 « بامتصاص » التباعد المتوافق مع نهاية السطر ، الخ . .) .
 إذا كانت `eof` صح فإن نداء `coln` هو خطأ .

3.4.2 - الدّخل : `readln, read`

عند تطبيقه على نص ، في الشأن معاينة (مثل `input`) ، فإن الإجراء المعرف مسبقاً `read` يسمح بقراءة :

- سمة (تلك الموجودة في النافذة ، التي ستنتقل فيما بعد إلى السمة التالية) ؛
 - عدد صحيح ، مع أو بدون علامة : يتم تجاهل التباعد ونهاية السطر السابقين للقيمة ؛
 تتوقف القراءة فور عدم وجود سمة من العدد الصحيح في النافذة .

مثال : مع `Var a, b : char; i, j, k : integer`

والمعطيات « bb - 12b0 ★ 1 ».

فإن (i, j, a, k, b)

تعطي
 $i = -12$
 $j = 0$
 $a = '★'$
 $k = 1$
 $b = ' '$

- عدد حقيقي ، مع أو بدون علامة : يتم تجاهل التباعد ونهاية السطر السابقتين للقيمة ؛
 تتوقف القراءة فور عدم وجود سمة من العدد الحقيقي في النافذة .
 إختصارات :

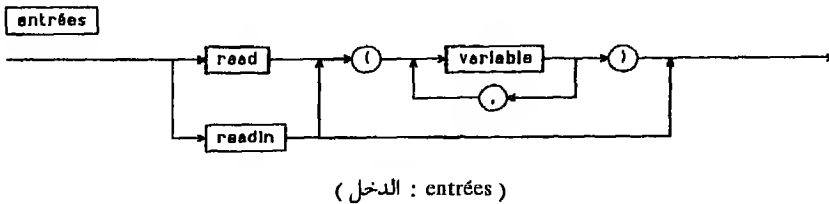
$read(v_1, v_2, \dots, v_n)$ هو مكافئ لـ
begin read $(v_1); read(v_2); \dots; read(v_n)$ **end**

إنتبه : يجب أن تكون القيمة المقروءة متساوقة بالنسبة للتعين مع المتغير المعين .
 ملاحظة : لا يسمح Read بقراءة قيمة من النوع تعداد (بولي مثلاً) ، دليل (Pointeur) ، مجموعة ، جدول ، سلسال أو سجل .

إن قراءة أعداد صحيحة أو حقيقية لا تسمح باستعمال الدالة eof ذلك لأنه يجري « تلبيس » سجل السمات بتكوين مختلف .

إن الإجراء المعرف مسبقاً readln والمطبّق فقط على النص ، يعمل على وضع النافذة على بداية السطر التالي (يعني إذن تخطي نهاية السطر) ، في حال وجوده .
 إختصارات :

$readln(v_1, \dots, v_n)$ هو مكافئ لـ
begin read $(v_1); \dots; read(v_n); readln$ **end**



4.4.2 - الخرج : writeln, write

عند تطبيقه على نص ، في الشأن نتائج (مثل output) ، فإن الإجراء المعرف مسبقاً write يسمح بكتابة :

- سمة ؛ يعمل write (e) على كتابة قيمة e على شكل سمة (إنه write (e:1)) يعمل
write (e:n) على كتابة n-1 تباعد ، ومن ثم قيمة e على شكل سمة .

مثال : ('a', 'a': 2) write يعطي aa

- عدد صحيح ؛ يعمل write (e) على كتابة التمثيل العشري لـ e (إنه write (e:n) حيث
تتعلق n بنوعية الحاسب الآلي المستعمل) .

يعمل write (e:n) على كتابة التمثيل العشري لـ e ، مسبقاً بقدر ما يلزم من التباعد
للحصول في النهاية على n سمة . إذا تعدى عدد الأرقام (زائد العلامة -) n ، يكون
قد تم تجاوز حقل الكتابة .

مثال : (12: 2, -12: 4, 127:1) write يعطي 12- 12127

- عدد حقيقي ؛ يعمل write (e) على كتابة قيمة e على شكل عائم (إنه write (e:n) حيث
تتعلق n بنوعية الحاسب الآلي المستعمل) .

يعمل write (e: n) على كتابة قيمة e على شكل عائم ، مسبقاً بعدد من التباعد لـ n
سمة بالإجمال . إذا كان حقل الـ n سمة غير كافياً ، فإنه سيتم توسيعه .

يعمل write (e:n:d) على كتابة قيمة e على شكل ثابت ، مع d رقم للقسم الكسري من
الجزء العشري ؛ يسبق ذلك عدد من التباعد لـ n سمة بالإجمال . إذا كان حقل الـ n
سمة غير كافياً ، فإنه سيتم توسيعه .

- سلسال ؛ يعمل write (e) على كتابة الـ x سمة من السلسال e وذلك بالترتيب .

يعمل write (e: n) على كتابة الـ n أول سمات من السلسال e ، مسبقاً بقدر ما يلزم
من التباعد للحصول على حقل من n سمة ، في حال $x < n$.

- بولي ، يعمل write (e) على كتابة السلسال 'true' أو السلسال 'false' (بالحرف الصغير
أو الكبير) .

يعمل write (e: n) على توسيع الحقل إلى n سمة ، كما هي حال السلسال .

إختصارات : write (e₁, e₂, ... , e_n) هو مكافئ لـ

begin write (e₁); write (e₂); ...; write (e_n) end

ملاحظة : لا يسمح write بكتابة قيمة من النوع تعداد (ما عدا البولي) ، دليل
(pointeur) ، مجموعة جدول ، فقرة أو سجل .

إن الإجراء المعرف مسبقاً writeln والمطبق فقط على النص ، يعمل على إنهاء
السطر .

إختصارات : writeln (e₁, ..., e_n) هو مكافئ لـ

begin write (e₁); ...; write (e_n); writeln end

مثال : أكتب n نجمة على سطر

```
for x:= 1 to n do write (' * '); writeln
```

مثال : أكتب $n * p$ خط وصل على سطر ، بمجموعات من p مفصولة بعلاقة جمع (+) :

--- + --- + --- +

```
for i:=1 to n do begin
  for j:=1 to p do write('--'); write('++') end;
writeln
```

مثال : أعد نسخ المعطيات (c ، متغير سمة)

```
while not eof do begin
  while not eoln do begin
    read(c); unite(c) end ;
  readln; writeln end
```

5.4.2 - ترتيب الصفحات أو الإخراج (mise en page)

يُسبب الإجراء المعرف مسبقاً page بتخطي الصفحة ، عندما يكون سجل الخرج مطبوعاً على جهاز ضوئي (périphérique) ملائم .

ملاحظة : على بعض الحاسبات الآلية ، ومع بعض الأجهزة الضوئية للخروج ، تكون السمة الأولى لكل سطر غير مطبوعة ، لكن تفهم كسمة تحكم لتقديم الورق ؛ تطبق إذن بشكل عام المصطلحات التالية :

' ' (تباعداً) : إنتقال عادي إلى السطر .

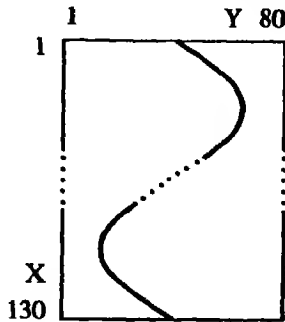
'0' (صفر) : أفقر سطرأ قبل الطباعة .

'1' : قفز الى الصفحة قبل الطباعة .

'+' : خطوة تقديم الورق .

6.4.2 - مثال : منحنى

يمكن الحصول على رسم تقريبي لمنحنى على الطباعة أو على الشاشة ؛ مثلاً لرسم $y = \sin(x)$ ، ضمن إطار من 80 عاموداً و130 سطرأ ، نطبع



عندنا إذن العلاقات ، بين الإحداثيات X (سطر) ، Y (عامود) و x, y الانطلاق

Y	1	→	80	X	1	→	130
py	- 1	→	+ 1	px	0	→	4 π environ

soit $Y = 1 + 79 \star (py + 1)/2$

$py = \sin(px)$

$px = 4 \pi (X/130)$

X تتغير من 1 إلى 130

التعريفات	المعجم
نتيجة = أكتب « نقطة في العامود Y » X من 1 إلى 130	Y (80...1) : رقم العامود X (130...1) : رقم السطر

$Y \leftarrow X$

3	y = entier (1 + 79 * (py + 1) / 2)
2	py = sin (px)
1	px = 4 * pi * X / 130

```

program courbe(output);
{ أرسم منحنى الجيب على الشاشة أو الطابعة }
const pi=3.14159;
var Y:1..80; { رقم العامود }
    X:1..130; { رقم السطر } ;
    px,py:real;
begin
  for X:=1 to 130 do begin
    px:=4.0*pi*X/130.0; py:=sin(px);
    Y:=trunc(1.0+79.0*(py+1.0)/2.0);
    writeln('.:Y) end
end.

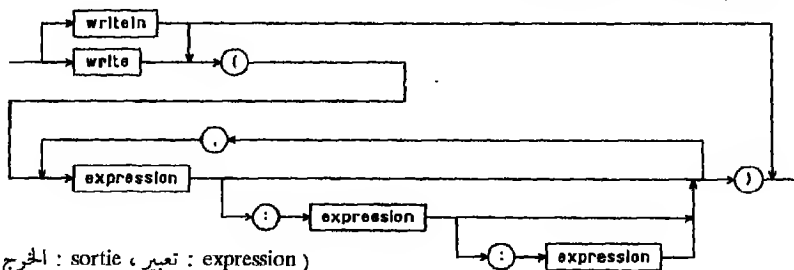
```

(courbe = منحنى)

ملاحظة : إن معالجة السجلات سيتم شرحها بشكل أوفر في 7.4

sorties

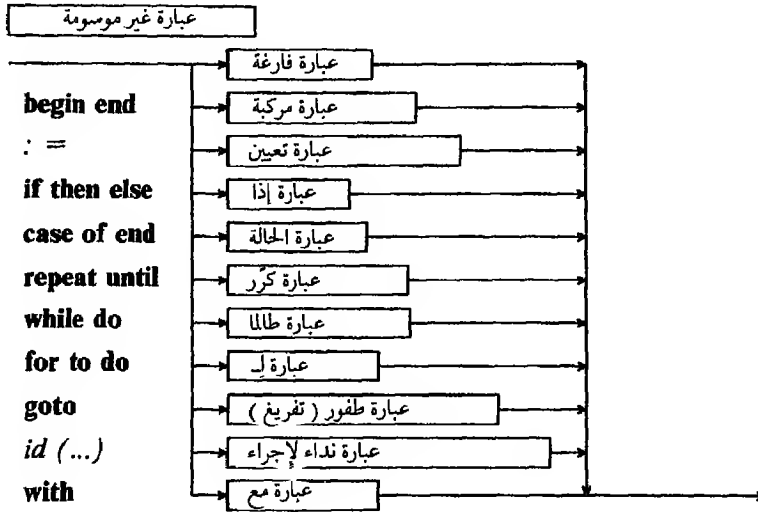
نحو الخرج (write أو writeln)



(sortie : الخرج ، expression : تعبير)

الفصل الثالث

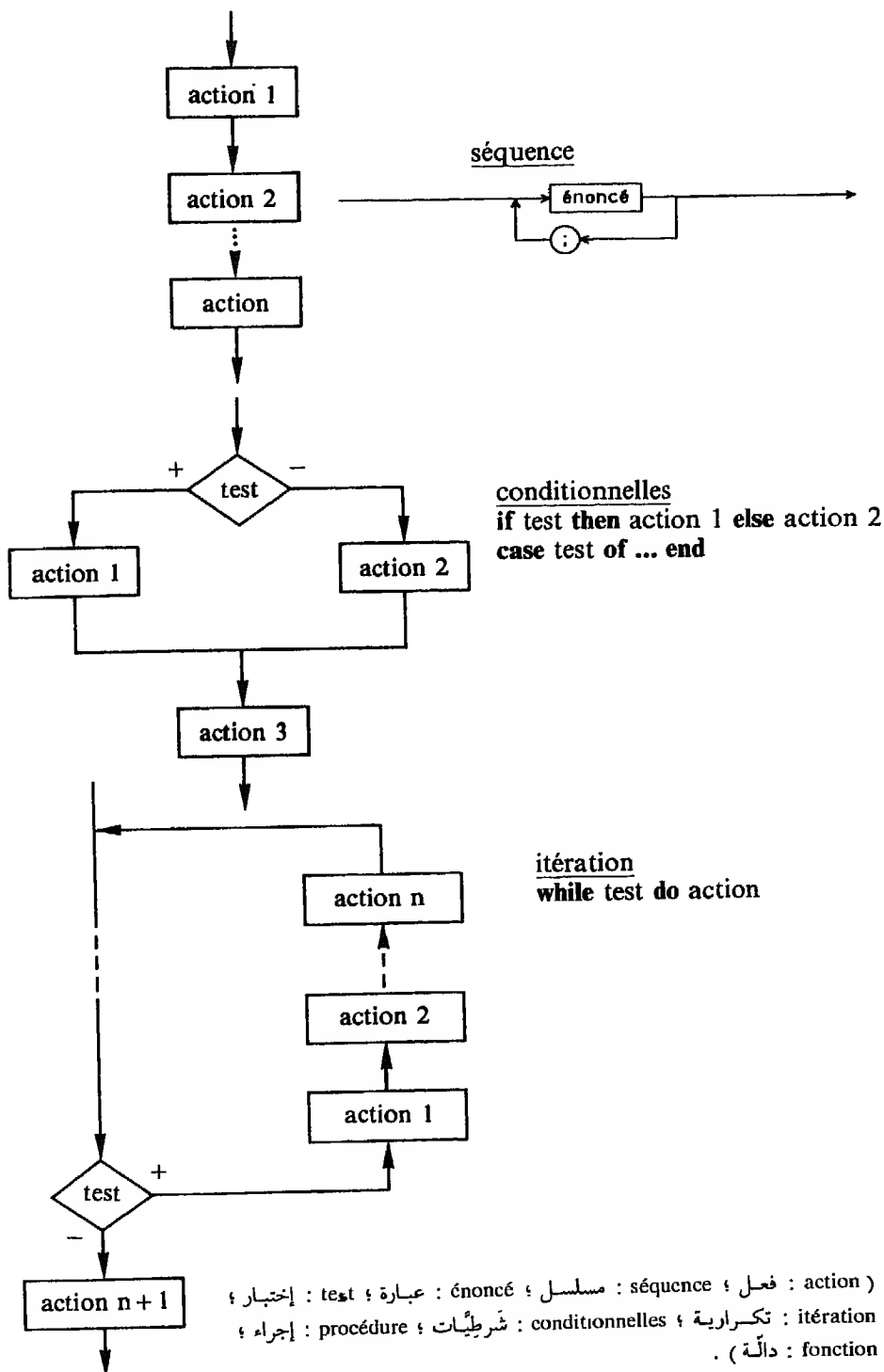
قواعد اللغة : معالجة الأدوات

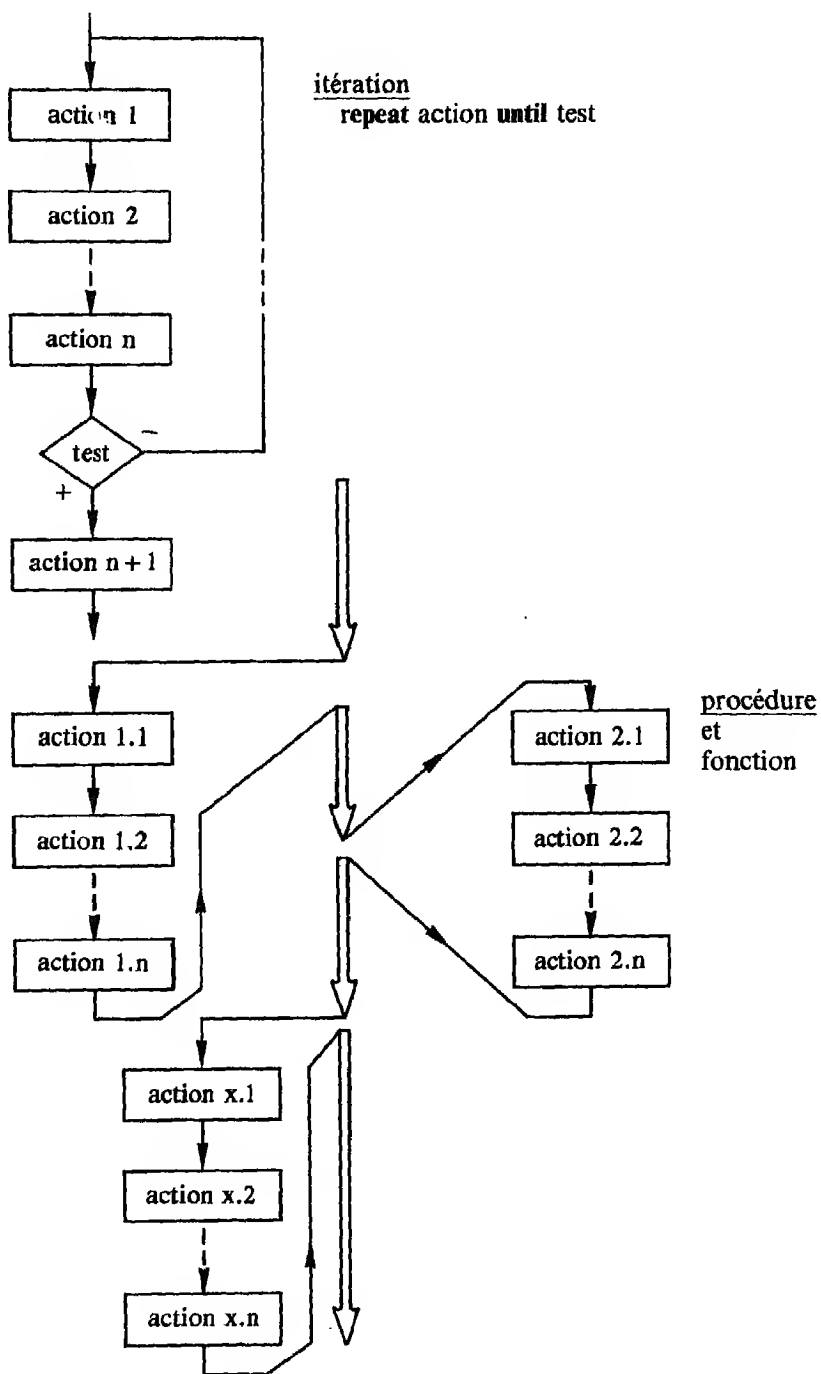


- تتألف فدر (bloc) برنامج من قسمين كبيرين :
- وصف للأفعال الواجب إتمامها ، مكتوبة على شكل عبارات ؛
- وصف للأدوات المعالجة من قبل هذه العبارات ، متمم بواسطة تصريحات وتعريفات :
- تصريح متغيرات ، تعريف أنواع وثوابت .

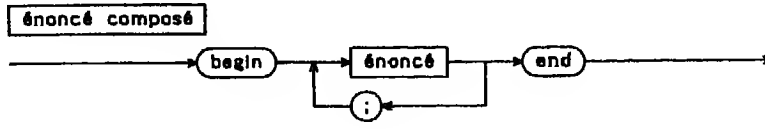
0.3 - عبارات (Enoncés)

إن سيل الحسابات الخاص بخوارزم أو برنامج لا يقرأ بالوصف الخطي : يوجد حلقات ، رجوع إلى الوراء وقفزات . تحتوي لغة الباسكال على تركيبات تحكم ، بأعداد صغيرة ، تسمح بإجراء هذا الوصف :





تُجمّع العبارة المركبة مسلسل من العبارات



(énoncé composé : عبارة مركبة) .

(إن الجسم هو كناية عن عبارة مركبة) .

يمكن إستعمال العبارة الفارغة في برنامج ، وبشكل عام بهدف التمكن من الإدخال الحر لـ « ؛ » :



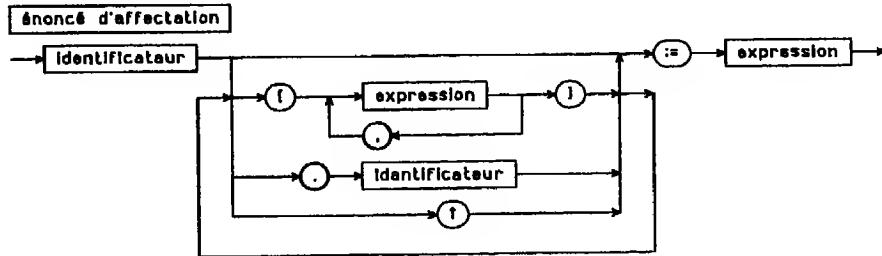
(énoncé vide : عبارة فارغة)

مثال : `begin a := 1 ; b := 2 ; end`

يتبين من خلال العبارة المركبة الواردة في هذا المثال بأنها تحتوي على 3 عبارات ،
إثنتين للتعين وواحدة فارغة .

1.3 - تعين ، تعبير (Affectation, expression)

إن العبارة الأكثر أساسية هي التعين ، الذي يسمح بتخصيص متغير ، أو مرُكّب
للمتغير ، بقيمة محسوبة حديثاً عن طريق تقييم لتعبير .



(énoncé d'affectation : عبارة تعين ؛ expression : تعبير ؛ identificateur : معرف)

إن قيمة التعبير ، من النوع T2 ، يجب أن تكون « متساوقة بالنسبة للتعين » مع
النوع T1 للمتغير المعين ، هذا ما يتم ثبوته فور تحقق إحدى الخصائص التالية :
أ - T1 و T2 هما نفس النوع ولا يحتويان على النوع سجل .

- ب - T1 من النوع حقيقي ، T2 من النوع صحيح (يوجد إذن تغيير أوتوماتي) . .
 ج - T1 و T2 هما نوعان ترتيبيان متساوقان (أنظر 7.3.2 (ب)) ، والقيمة من النوع T2 موجودة ضمن الفترة المحددة من قبل T1 .
 وكذلك)
 د - T1 و T2 هما نوعا مجموعتين متساوقان ، والمجموعة المعينة يمكن أن تحوي القيمة .
 هـ - T1 و T2 هما نوعا سلسلسال متساوقان) .
 مثال :

```

type T=array [char] of boolean;
var a:integer;      b:real;      c:char;
    x:T;           y:'0'..'9';   y:array [boolean] of T;

a:= 3              تعيين سليم (أ)
b:= a              تعيين سليم (ب)
c:= y              تعيين سليم (ج)
x [chr (ord (y) + ord ('a') - ord ('0'))):= true  تعيين سليم ;
a:= b              تعيين خاطيء (ب)
y:= '+'            تعيين خاطيء (ج)
c:= a              تعيين خاطيء (أ)
y [false]:= x      تعيين صحيح (أ)
    
```

يمكن أن نعين عدد صحيح لعدد حقيقي ، لكن العكس غير صحيح بتاتا (إن هذا منطقي ذلك لأنه سيوجد في هذه الحالة فقدان للدقة (الأعداد العشرية) غير متحكم به) .

بما أن المتغير يفقد قيمته القديمة عند إجراء التعيين ، يجب استعمال متغير مؤقت لإجراء عملية تبديل للقيم :
 لتبديل قيم u و v نكتب :

t:= u; u:= v; v:= t

إن القيمة الأولية للمتغير هي غير محددة ؛ سيكون من الخطأ استعمالها قبل أية عملية تعيين .

1.1.3 - تحليل

إن تعريفاً بسيطاً مثل $u = v + 1$ من التحليل ، يتطابق مع عبارة التعيين $u := v + 1$ من البرنامج .

إن التعريف بالتكرار (الثنية إلى الوراء) للمتسلسلة u :
 $0 = u$ وأول $u = f(u, \dots)$

تتم ترجمته إلى لغة الباسكال بـ

$$u := 0 ; \dots u := f(u, \dots)$$

مثال : لنفترض أننا نريد حساب الحدود الأولى من متسلسلة Fibonacci المعروفة بـ

$$f_0 = 0, f_1 = 1 \text{ et } f_i = f_{i-1} + f_{i-2} \\ (f_2 = f_1 + f_0 = 1, f_3 = f_2 + f_1 = 2, f_4 = 3, \dots).$$

المعجم	التعريفات
i - (صحيح)	نتيجة = أكتب ' ' , 'f' , 'i' , ' ' من 2 إلى 10
f_i : i - (صحيح) F -	أول 1 = F
f_{i-1} : i - (صحيح) G -	أول 0 = G
$f \leftarrow i$	$\overline{F} = F + \overline{G}$ $\overline{G} = \overline{F}$

لقد تم إدخال متسلسلة وسيطة G « لتخزين » القيم f_{i-2} المفقودة كلما تقدمت العمليات الحسابية .

بالفعل فإن كتابة $f = f + f$ شيء لا يمكن ترجمته مباشرة إلى لغة الباسكال !

F هي إذن متسلسلة الـ f_i و G متسلسلة الـ f_{i-1} .

كذلك فإننا بحاجة إلى القيمة القديمة لـ F لكي نتمكن من تعريف G ، وإلى القيمة القديمة لـ G لتعريف F ؛ هذا ما يؤدي إلى إدخال معرف مساعد t :

$$\begin{array}{l|l} 2 & F = \overline{F} + \overline{G} \\ 3 & G = t \\ 1 & T = \overline{F} \end{array}$$

```

program Fibonacci(output);
{ حساب الحدود الأولى من متسلسلة Fibonacci }
var i:integer;
    F,G,t:integer; {F=f(1), G=f(1-1),
                  {متغير مساعد 1}
begin
  G:=0; F:=1;
  for i:=2 to 10 do begin
    t:=F; F:=F+G; G:=t;
    writeln('f(',i,',')= ',F)
  end
end.

```

ملاحظة : إن التفكير على مستوى « أكثر تجزئاً » من عبارات برنامج تساعدنا هنا على الحصول وبسرعة على حل سليم .

2.1.3 - تعبيرات (Expressions)

كونه مكوناً من متاثرات (opérandes) (متغيرات ، ثوابت ، ...) ومؤثرات (opérateurs) ، يسمح التعبير بتحديد قاعدة للحساب .

يجب أن يكون لكل من المتاثرات قيمة (ترتيبية ، حقيقية ، أو مجموعة) ؛ إن الإستعمال في تعبير ، المتغير ذي قيمة غير محددة يصبح غلطاً .

تترسّخ الأسبقيات الخاصة بالمؤثرات تبعاً للفتات الأربعة التالية والمعطية ضمن الترتيب للأسبقيات المتناقصة :

- المؤثر not (لا) ؛

- مؤثرات الضرب (و) (and , / , * , mod , div) ؛

- مؤثرات الجمع (+ ، - ، أو) (or) والعلامات (+ ، -) ؛

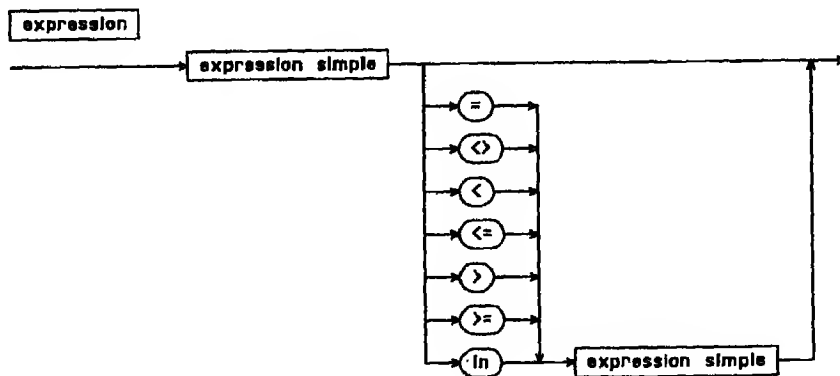
- مؤثرات العلاقة (= ، < ، > ، <= ، >= ، < >) (in ، >) .

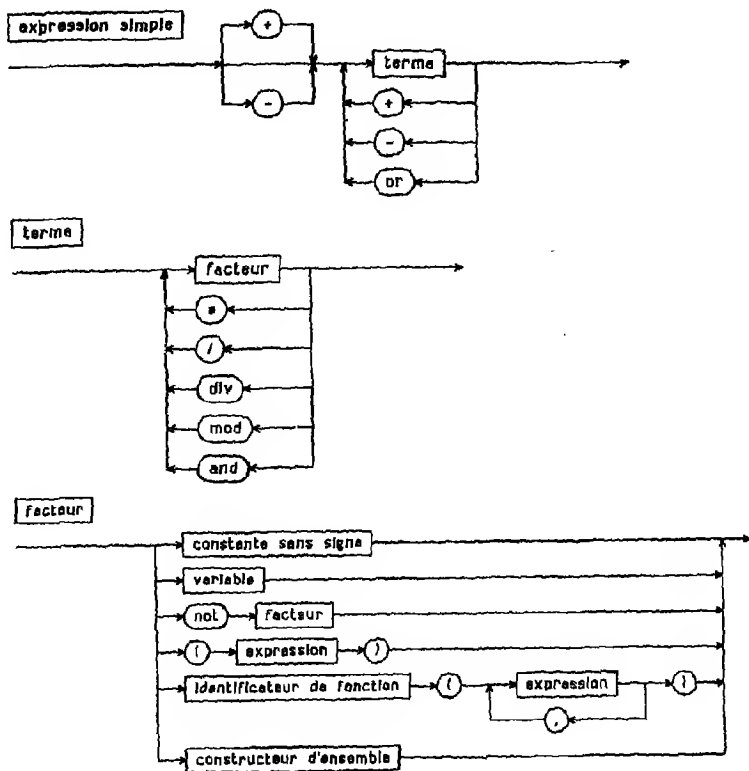
ضمن الأسبقية المتساوية ، يتم التقييم من اليسار إلى اليمين . يمكن دائماً تغيير ترتيب التقييم بواسطة الأقواس (()) .

أمثلة :

$a+b-c+d$	تعني	$((a+b)-c)+d$
$a+b*c-a/x$	تعني	$(a+(b*c))- (a/x)$
$a/b/c$	تعني	$(a/b)/c$ soit $a/(b*c)$
$a/b*c$	تعني	$(a/b)*c$ soit $(a*c)/b$

يجزّأ التعبير الى عوامل ، حدود وتعابير بسيطة على الشكل التالي :





تعبيرات بسيطة
 $p \text{ or } q$
 $x+y$
 $\sim x$
 $i*j+1$

تعبيرات
 $x=1.5$
 $p=q$
 $p=q \text{ and } r$
 $c \text{ in teinte}$

إن نحو التعبير هو تكراري ؛ إنه يحتوي على عامل الذي بدوره يحتوي على تعبير .

3.1.3 - مؤثرات حسابية :

الرمز	العملية	نوع المتأثرات	نوع النتيجة
-	طرح	صحيح وصحيح	صحيح
+	جمع	صحيح وحقيقي	حقيقي
*	ضرب	حقيقي وصحيح حقيقي وحقيقي	
/	قسمة	صحيح وصحيح صحيح وحقيقي حقيقي وصحيح حقيقي وحقيقي	حقيقي
div	قسمة صحيحة	صحيح وصحيح	صحيح
mod	مقياس	صحيح	صحيح

ملاحظة : كذلك نستعمل + ، - ، * كمؤثرات على المجموعات .

x / y هي دائما من النوع حقيقي وتكون غلط إذا كانت $0.0 = y$

$x \text{ div } y$ هي غلط إذا $0 = y$

$i \text{ mod } j$ هو غلط إذا $0 \leq j$

في حال كانت $i \geq 0$ و $j < 0$ يصبح :

$$i = (i \text{ div } j) * j + i \text{ mod } j$$

إن حاصل قسمة i بـ j هو $i \text{ mod } j$

تُطبق المؤثرات ذات المتأثر الواحد + (تطابق) و - (إنقلاب العلامة) على الأعداد الصحيحة والحقيقية ؛ تكون النتيجة من نوع المتأثر .

يطلق إسم التعبير المختلط على التعبير الذي تتم فيه تغييرات أوتوماتية من النوع صحيح إلى النوع حقيقي : $1 + 2.0/4$ ($1 \rightarrow 1.0$, $4 \rightarrow 4.0$)

4.1.3 - مؤثرات بولية

الرمز	العملية	نوع المتأثرات	نوع النتيجة
(و)	عطف	بولي وبولي	بولي
(أو)	تفرُّق		
(لا)	نفي	بولي	

p and q

$p \backslash q$	خطأ $false$	صح $true$
p		
$false$	$false$	$false$
$true$	$false$	$true$

p or q

$p \backslash q$	$false$	$true$
p		
$false$	$false$	$true$
$true$	$true$	$true$

p	not p
$false$	$true$
$true$	$false$

إنها المؤثرات التقليدية المستعملة في جَبْر بُول . هناك مؤثرات أخرى تتفرع عنها (أو المقتصرة (ou exclusif) ، لا - و ، ...) .

5.1.3 - مؤثرات العلاقة

الرمز	العملية	نوع المتأثرات	نوع النتيجة
=	مُساوٍ	كل نوع بسيط أو كل سلسال	بولي
< >	مُختلفٌ عن		
<	أقل من		
>	أكبر من		
< =	أقل أو مساوٍ		
> =	أكبر أو مساوٍ		

على المتأثرات أن تكون من أنواع متساوقة (أنظر 7.3.2) ، = ، > ، < ، = ، > ، < = (أنظر 1.3) ؛ و < > تطبق على الأنواع دليل (Pointeur) (أنظر 6.3) .

6.1.3 دوال معرفه مسبقاً

إلى هذه المؤثرات ، تضاف مؤثرات مقدّمة على شكل دوال ؛ تكون هذه الدوال حسابية ، ترتيبية ، للنقل ، أو بولية .

دوال حسابية (Fonctions arithmétique)

على متأثر x من النوع على السواء حقيقي أو صحيح ، تعطي هذه الدوال نتيجة « حقيقي » ، ما عدا sqr و abs حيث تكون للنتيجة نوع المتأثر x .

abs (x)	تحسب القيمة المطلقة (حقيقي أو صحيح) لـ x
sqr (x)	تحسب مُرَبَّع (حقيقي أو صحيح) x
sin (x)	تحسب جيب الزاوية x (x بالراديان)
cos (x)	تحسب جيب التمام للزاوية x (x بالراديان)
arc tan (x)	تحسب القيمة بالراديان لقوس ظل x
exp (x)	تحسب e^x
ln (x)	تحسب اللوغاريتم الأعلى لـ x ، في حال $x > 0$
sqrt (x)	تحسب الجذر التربيعي غير السالب لـ x ، في حال $x > 0$

(يكتب كذلك الرفع إلى قوة x^y : $\exp (y * \ln (x))$)

دوال ترتیبیّة (fonctions ordinales)

$\text{succ}(x)$: هو تعبير من نوع ترتيبي ، والنتيجة من نفس النوع ؛ إنها القيمة التي يكون عددها الترتيبي مباشرة أكبر من ذلك الخاص بـ x (إذ وُجِدَ)
 $\text{pred}(x)$: هو تعبير من نوع ترتيبي ، والنتيجة من نفس النوع ؛ إنها القيمة التي يكون عددها الترتيبي مباشرة أقل من ذلك الخاص بـ x (إذا وُجِدَ)
 $\text{ord}(x)$: تعطي العدد الترتيبي ، من نوعٍ صحيح ، للمتأثر x ؛ x هو تعبير من نوع ترتيبي .

$\text{chr}(x)$: تعمل على المطابقة للتعبير x ، من النوع صحيح ، السمة التي يكون عددها الترتيبي هو قيمة x (إذا وُجِدَ) . إن العلاقة $\text{chr}(\text{ord}(c)) = c$ هي صحيحة بالنسبة لكل سمة c .

دوال النقل (Fonctions de transfert)

$\text{trunc}(x)$: كُون x من النوع حقيقي ، فَإِنَّا نحصل على « القسم الصحيح » :

القسم الصحيح لـ x إذا $x \geq 0$

- القسم الصحيح لـ $\text{abs}(x)$ إذا $x < 0$

أمثلة : $\text{trunc}(17.986) = 17$

$\text{trunc}(-4.3) = -4$

$\text{round}(x)$: كَوْن x من النوع حقيقي ، فإننا نحصل على « العدد الصحيح الأقرب » :

$x \geq 0$ إذا $\text{trunc}(x + 0.5)$

$x < 0$ إذا $\text{trunc}(x - 0.5)$

$\text{round}(17.986) = 18$

أمثلة : $\text{round}(-4.3) = -4$

$\text{round}(3.5) = 4$

$\text{round}(-3.5) = -4$

مثال : $\text{var } i : \text{integer} ; r : \text{real}$

$r := 9.6 ; i := (\text{trunc}(r) + \text{round}(r)) \bmod 2$

i تساوي 1

دوال بوليّة (Fonctions booléennes)

$\text{odd}(x)$ تساوي true (صحّ) إذا كان التعبير x من النوع الصحيح هو مفرداً ، وإلاّ فإنها

تساوي false (خطأ) (إن $\text{odd}(x)$ هي $\text{abs}(x) \bmod 2 = 1$).

تطبّق الدوال eof و eoln على السجلات والنصوص :

$\text{eof}(f)$ تعني بأن النافذة قد وصلت إلى نهاية السجل f

$\text{eoln}(f)$ تعني بأن النافذة قد وصلت إلى نهاية سطر في النص f

إذا الغي الوسيط f فهذا يعني بأن المقصود هو السجل input

مثال : إحسب الخطوط الخاصة بحساب المثلثات والجذر التربيعي لقيمة معطية .

البرنامج

```
program lignes(input,output):
```

```
{ خطوط المثلثات والجذر التربيعي }
```

```
var x:real; { قيمة معطية بالراديان }
```

```
begin
```

```
  read(x);
```

```
  writeln('x':12, 'sin':12, 'cos':12);
```

```
  writeln(x:12:5, sin(x):12:5, cos(x):12:5);
```

```
  writeln;
```

```
  writeln('tg':12, 'cotg':12, 'racine':12);
```

```
  writeln(sin(x)/cos(x):12:5, cos(x)/sin(x):12:5,
```

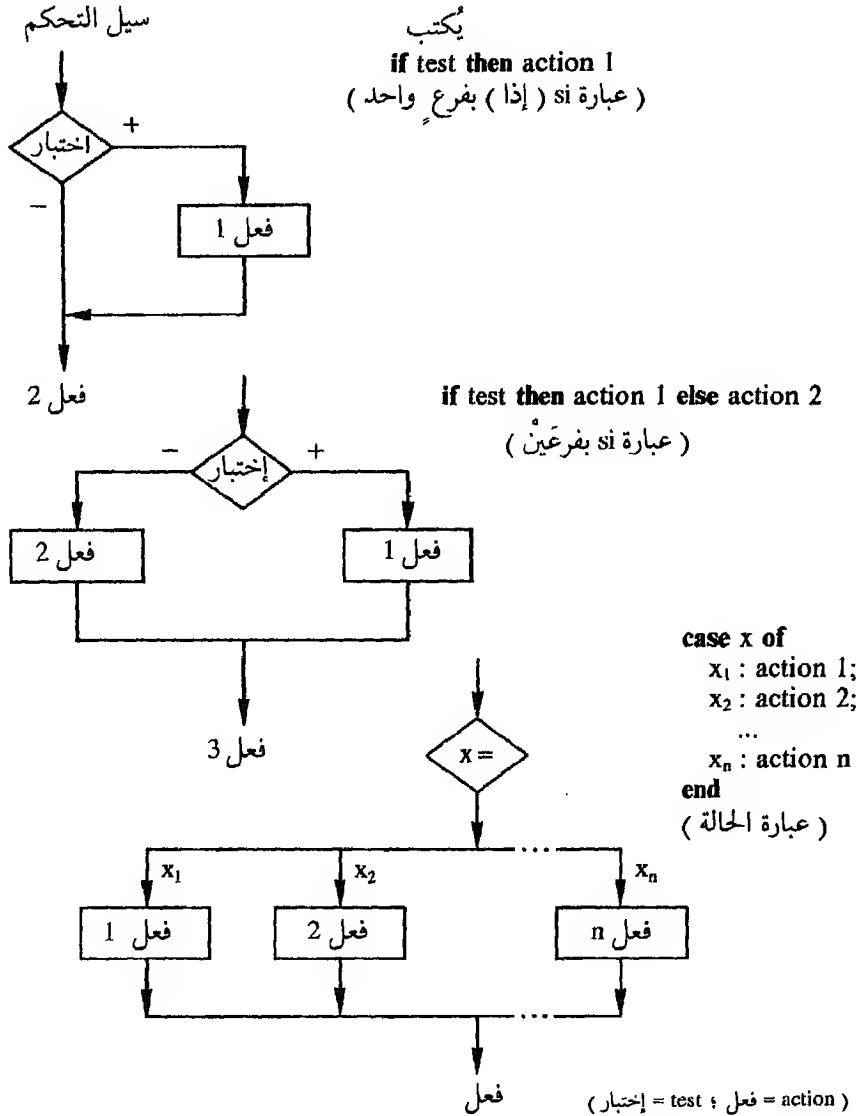
```
          sqrt(x):12:5)
```

```
end.
```

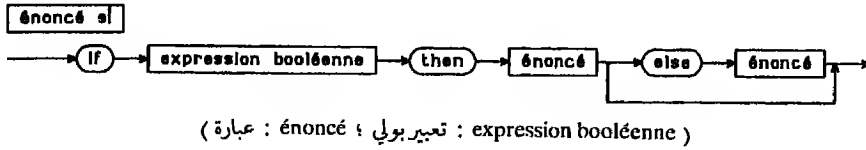
يعطي هذا البرنامج النتيجة :

x	\sin	\cos
0.10000	0.09983	0.99500
tg	$cotg$	$racine$
0.10033	9.96664	0.31623

2.3 - الشَّرَطِيَّات



1.2.3 - عبارة إذا (Enoncé si)



إذا كان للتعبير البولي القيمة true (صح) ، فإن العبارة التي تلي then تُنفذ وحدها . أما إذا كانت قيمة التعبير false (خطأ) ، فإن العبارة التي تلي else ، إذا وُجِدَتْ ، تُنفذ وحدها .

مثال : `if x<1.5 then z:=x+y else z:=1.5`

مثال : `if j=0 then
 if i=0 then writeln('indefini')
 else writeln ('infini')
else writeln (i div j)`

تكون كل else مربوطة بال then السابقة الأقرب والتي ليست بعد مربوطة بـ else . إذا وُجِدَتْ عدة عبارات في قسم then أو قسم else ، نُكوِّن عبارة مركبة :

`if C then begin S1; S2;...; Sn و else begin s1; s2;...; sx end`

ملاحظة : لا يجب الخلط بين `if B then S1 else S2` و `if B then S1; S2`

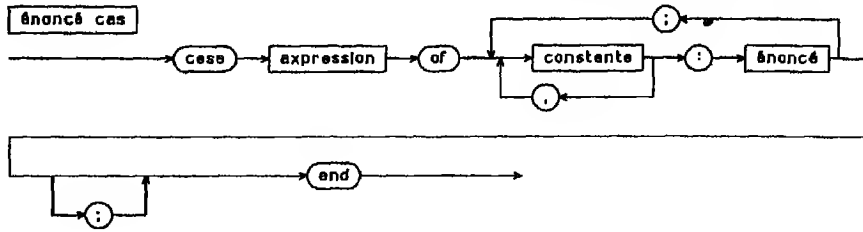
مثال : نريد ترتيب قيمتين a و b ، بشكل أن $a < b$:

`if a > b then begin c:= a; a:= b; b:= c end`

مثال : لنفترض معنا وقت معين (على شكل ساعة (h) ، دقيقة (mn) ، ثانية (s)) ، إجمع إلى هذا الوقت 2 د 10 ث (دقيقتين وعشر ثواني) .

```
program heure(input,output):
{ إجمع 2 د 10 ث لوقت معطى }
var h,mn,s:integer;  r:integer;
begin
  read(h,mn,s);
  s:=s+10;
  if s>=60 then begin s:=s-60; r:=1 end else r:=0;
  mn:=mn+2+r;
  if mn>=60 then begin mn:=mn-60; r:=1 end else r:=0;
  h:=(h+r) mod 24;
  writeln (h,mn,s)
end.
```

2.2.3 - عبارة الحالة (Enoncé cas)



(énoncé : عبارة ؛ expression : تعبير ؛ constante : ثابت)

يتم تقييم التعبير : قيمته تؤدي إلى تنفيذ العبارة الموافقة لثابت الحالة الذي يعبر عن هذه القيمة . يجب أن تكون كل ثوابت الحالة مختلفة ومن نفس النوع الترتيبي الذي للتعبير . إذا لم يكن أي من ثوابت الحالة مساوياً لقيمة التعبير ، يُعدّ هذا غلطاً . إنَّ « ؛ » قبل الـ end هو اختياري .

مثال :

```
case caractereLu of
  '+' : x:=a+b;
  '-' : x:=a-b;
  '*' : x:=a*b;
end
```

(caractere lu : السمة المقروءة)

أو كذلك مع التصريح (*plus, moins, fois*) *var opérateur :*

```
case opérateur of
  plus : x:=x+y;
  moins : x:=x-y;
  fois : x:=x*y;
end
```

(opérateur : مؤثر ؛ plus : + ؛ moins : - ؛ fois : *)

```
if a<b then c:=a else c:=b
case a<b of
  false : c:=b;
  true : c:=a
end
```

التعبير
يُكتب كذلك :

مثال : إطبعم العمل الذي يجب القيام به في كل يوم من الأسبوع :

```

program semaine(input,output);
type jour=(lundi,mardi,mercredi,jeudi,vendredi,samedi,
            dimanche);
var j:jour;      k,n:integer;      {j=ord(jour)}
begin
  read(n);
  j:=lundi; for k=1 to n do j:=succ(j);
  case j of
    lundi,mardi,mercredi,jeudi: writeln('au travail');
    vendredi: writeln('au travail, pour 7 heures');
    samedi: writeln('on sort ce soir');
    dimanche: writeln('on recupere')
  end
end.

```

(semaine : أسبوع ؛ lundi : الإثنين ؛ mardi : الثلاثاء ؛ mercredi : الأربعاء ؛ jeudi : الخميس ؛
 vendredi : الجمعة ؛ samedi : السبت ؛ dimanche : الأحد ؛ au travail : إلى العمل ؛ pour 7 heures :
 لسبع ساعات ؛ on sort ce soir : نخرج هذا المساء ؛ on recupere : نسترد قوتانا) .

ملاحظة : عندما يأخذ التعبير الذي يلي case ، قيمة خارجة عن قائمة ثوابت
 الحالة ، فهذا غلط . في بعض الآلات ، نستعمل تمديداً (خارج إطار القاعدة
 AFNOR) للغة الباسكال : عبارة مُدخلة بـ else أو otherwise تُجمَع كل الحالات غير
 المُعالَجة صراحة :

```

case op of
  '+':u:=u+v;
  '-':u:=u-v;
else:writeln('operateur inconnu')
end

```

(opérateur inconnu : مؤثر مجهول)

وإلاّ نتجنب المشكلة عن طريق التحقق المُسبق لقيمة التعبير :

```

if (op='+') or (op='-') then
  case op of
    '+':u:=u+v;
    '-':u:=u-v
  end
else
  writeln('operateur inconnu')
end

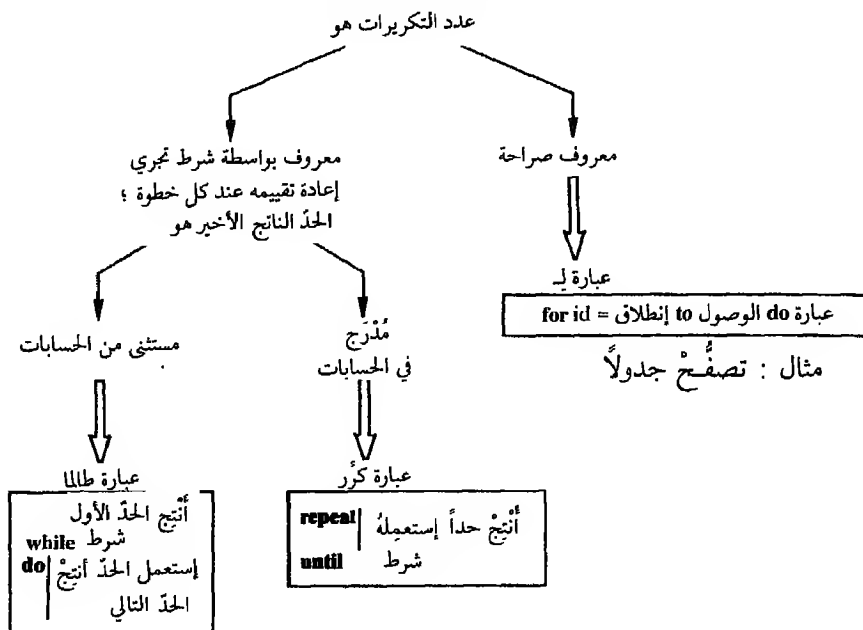
```

أو بطريقة أشمل :

if op in ['+', '-'] then ... Cf. 4.4).

3.3 - طريقة تكرارية (Itération)

تعرض لغة الباسكال ثلاثة طرق لتنفيذ عبارة بشكل مكرر :

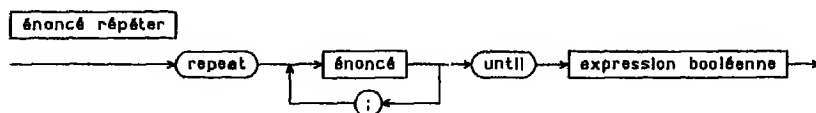


الإقصار (Exclusion) : « توقيف معلوماتي » ، لا تنفع المعطية الأخيرة إلا للإشارة الى نهاية الحسابات .

التضمين (Inclusion) : البحث عن الحذ الأول لتسلسلة والذي يُحقّق خاصية معينة .

لا تغطّي هذه الكتابات الثلاثة كل الحاجات . يجب وبوجه خاص أن تتمكّن التكرارية من التوقف فور ظهور غلط أو ظهور عدم ترابط في المعطيات : إن عبارة الطفور go to (branchement) تسمح بهذا التوقف المفاجيء .

1.3.3 - عبارة كرّر (Enoncé répéter)



(expression booléenne : تعبير بولي ؛ énoncé : عبارة) .

يتم تنفيذ متسلسلة العبارات بطريقة مكررة حتى يأخذ التعبير البولي القيمة true (صح) يتم تنفيذ متسلسلة العبارات مرة واحدة على الأقل ، ذلك لأنه يتم إختبار التوقف بعد التنفيذ .

إنتبه : يجب أن تستطيع متسلسلة العبارات تغيير قيمة التعبير البولي (وإلا فإن الحلقة لا تنتهي) :

repeat V = f (V) ; cond = g (V) **until** cond

(cond) (إختصار لـ condition = شرط)

مثال : numdec (عدد عشري)

لنفرض أننا نريد تحويل قيمة رقمية V صحيحة موجبة وأقل من 99999 إلى تمثيلها العشري (على شكل سمات) .

إن للقيمة V التمثيل العشري «t₄ t₃ t₂ t₁ t₀» حيث يكون t₀ رقم الآحاد ، t₁ رقم العشرات ، t_i الرقم المعين بالمعامل 10ⁱ . إن القيمة التي يمثلها t_i (لنفرض (ord (t_i) - '0') هي إذن :

$$\begin{aligned} t_0 &= \text{chr} (\text{ord} ('0') + v \bmod 10) \\ t_1 &= \text{chr} (\text{ord} ('0') + (v \div 10) \bmod 10) \\ &\dots \\ t_4 &= \text{chr} (\text{ord} ('0') + (v \div 10000) \bmod 10) \end{aligned}$$

هذا ما يؤدي إلى تعريف متسلسلتين تكراريتين (مثنى إلى الورا) Suites (T récurrentes و R) .

$$\begin{cases} T_i = R_{i-1} \bmod 10 & \text{متسلسلة قيم الأرقام العشرية} \\ R_i = R_{i-1} \div 10 & \text{متسلسلة « الحواصل »} \\ R_{-1} = v \end{cases}$$

مع $t_i = \text{chr} (\text{ord} ('0') + T_i)$

نَحسب هكذا الأرقام المتتالية (t₀ إلى t₄) بالترتيب المعاكس لإستعمالهم الطبيعي : نستعمل جدولاً لتخزينهم . يبدأ الحساب مع R₁ = V وينتهي عند n بشكل أن R_{n-1} = 0 أي '0' t_k بالنسبة لـ K ≥ n .

```
program numdec(input,output);
{ تحويل قيمة رقمية صحيحة إلى تمثيلها العشري على شكل سمات }
var v:integer;          { القيمة المطلوب تحويلها }
    t:array[0..4] of char; { التمثيل العشري }
    ... , t[3] , t[4] { أحاد [4] t ، عشرات [3] t ، ... }
```



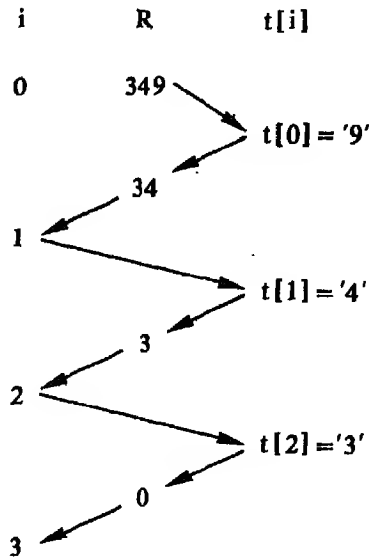
```

R:integer;      { متسلسلة الحواصل المطلوب تحويله }
i:integer;      { دليل الرقم المحسوب ، في t }
j:integer;      { لتصفح الجدول }
begin
  read(v); R:=v; i:=0;
  repeat
    t[i]:=chr(ord('0')+R mod 10);
    R:=R div 10;
    i:=i+1
  until R=0;
  for j:=i-1 downto 0 do write(t[j]); writeln
end.

```

نستعمل عبارة كرر لأن آخر $t[i]$ محسوب هو رقم غير الصفر ، فإذا كان مُعَبَّر ؛ وكذلك لأنه يوجد على الأقل رقم يجب حسابه $t[0]$ ، فإذا كان على الأقل تنفيذ واحد للحلقة .

خلال عملية الحساب ، عندنا مثلاً بالنسبة لـ $V = 349$



لتحويل قيمة رقمية الى تمثيلها الثنائي (binaire) (في القاعدة 2) أو الثماني (octale) (في القاعدة 8) ، يكفي أن نستبدل في التكرارية الثابت 10 بـ 2 أو بـ 8 .
 للتحويل الى التمثيل السادس عشري (hexadécimale) (في القاعدة 16) ، نستبدل 10 بـ 16 وتتم عملية تحويل قيمة الرقم (0 إلى 15) إلى سمة ('9', 'A', 'B', 'C', 'D', 'E', 'F')
 عن طريق التقسيم (indexation) للجدول C الحاوي للسمات ('0', '1', ..., '8')

(T : array [0..15] of char et $T[0] := '0', T[1] := '4', \dots, T[10] := 'A', \dots, T[15] := 'F'$) :

repeat $t[i] := C[R \bmod 16]$

2.3.3 - عبارة طالما (Enoncé tant que)



تنفذ العبارة بطريقة مكررة طالما يأخذ التعبير البولي القيمة true (صح) .
لا تنفذ العبارة بتاتاً إذا كان للتعبير البولي القيمة false (خطأ) عند الإنطلاق .
إنّبه : يجب أن تستطيع العبارة تغيير قيمة التعبير البولي (وإلا فإن الحلقة لا تنتهي)

while cond do $\begin{cases} V = f(\bar{V}) \\ \text{cond} = g(V) \end{cases}$

(cond) (إختصار لـ condition = شرط)

إذا كان يجب ظهور عدة عبارات في قسم **do** ، نُكوّن منها عبارة مركّبة :

while c do begin énoncé; ... énoncé end

العبارة **while b do E** هي مكافئة لـ :

begin if b then repeat E

until not (b)

end

لا يجب أن ننسى إنتاج الحدّ الأول قبل بدء تنفيذ عبارة **while** :

$V = V_0$; **while cond (V) do** $V = f(\bar{V})$

مثال : binnum (عدد بالتمثيل الثنائي)

لنفرض أننا نريد التحويل إلى قيمته الرقمية ، عدد موجب صحيح معطي بالتمثيل الثنائي (متسلسلة من السمات ، ممثلة القيمة في القاعدة 2) .

إن للقيمة V التمثيل الثنائي « $t_n \dots t_4 t_3 t_2 t_1 t_0$ » حيث أن t_i مُلحق بالمعامل 2^i . إذا سمّينا T_i القيمة المطابقة للسمة t_i ($T_i = \text{ord}(t_i) - \text{ord}('0')$) ، نحصل على :

$$v = T_n 2^n + \dots + T_3 2^3 + T_2 2^2 + T_1 2^1 + T_0 2^0$$

$$= (((\dots (T_n 2 + T_{n-1}) 2 + T_{n-2} \dots + t_1) 2 + t_2) 2 + t_1) 2 + t_0$$

(إن هذا هو إنسباط هورنر (développement de Horner) لتعدد الجذور) .
إذن تعرّف القيمة V بواسطة متسلسلة تكرارية :

$$v = v_0 \quad \begin{cases} v_i = v_{i+1} \star 2 + T_i \\ v_{n+1} = 0 \end{cases}$$

```

program binnum(input,output);
{ التحويل إلى قيمة رقمية لقيمة ثنائية }
var t:char;      { رقم ثنائي مُعناد }
    v:integer;   { قيمة }
begin
  v:=0;
  read(t);
  while (t='0') or (t='1') do begin
    v:=v*2+ord(t)-ord('0');
    read(t);
  end;
  writeln (v)
end.

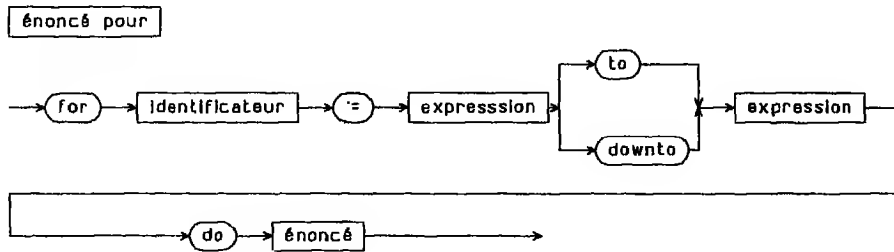
```

إنتاج العنصر الأول ←
إستعمال العنصر الأخير المُنتج ←
إنتاج العنصر التالي ←

نستعمل عبارة while لأن آخر t مُنتج ليس رقماً من العدد .

3.3.3 - عبارة لِ (Enoncé pour)

لا تستعمل العبارة « لِ » إذا كان عدد التكريرات معروفاً . إن فائدتها تكمن في أنها تقدم « عدّاداً » ، أمر كثير المنفعة مثلاً لتصفّح جدولاً .



(identificateur : معرف ؛ expression : تعبير ؛ énoncé : عبارة)

إن المتغيّر والمسمى متغير التحكّم ، يأخذ بالتتابع كل القيم من تعبير الإنطلاق حتى تعبير الوصول ؛ لكل قيمة ، يتم تنفيذ العبارة .
- مع to ، يتم الإنتقال الى القيمة التالية بواسطة succ

- مع **downto** ، يتم ذلك الانتقال بواسطة **pred**
- يكون متغير التحكم من نوع ترتيبي ؛ يجب أن تكون قيم الإنطلاق والوصول من نوع متساوق مع هذا النوع .
- يجب أن يكون متغير التحكم بسيطاً (معرّف ، غير مركّب) وموضعياً (مصرّح في قسم تصريحات متغيرات الفدرة الفعّالة) .
- بعد إنتهاء العملية التكرارية ، تكون قيمة متغير التحكم غير محدّدة .
- لا يجب بتاتاً التغير المباشر لمتغير التحكم .
- إذا كانت قيمة الإنطلاق أكبر من قيمة الوصول (مع **to** ، أو أقل مع **downto**) ، فإن العبارة لا تنفّذ .
- إذا وضعت هذه القيود على جدّة ، فإن العبارة :

for v : = e1 to e2 do E

مكافئة لـ :

```
begin t1:=e1; t2:=e2;
  if t1<=t2 then begin
    v:=t1; E;
    while v<>t2 do begin
      v:=succ(v); E end
    end
  end
end
```

حيث تكون $t1$ و $t2$ متغيرات مساعدة لا تظهر في مكان آخر من البرنامج .

مثال : Trigo (الخطوط الخاصة بحساب المثلثات)

لنفرض أننا نريد طبع جدولاً للخطوط الخاصة بحساب المثلثات ، جيب (sinus) وجيب التمام (cosinus) على الشكل :

I DEGRES I SIN I COS I							
I	0	I	0.000	I	1.000	I	90
I	1	I	0.017	I	1.000	I	89
I	2	I	0.035	I	0.999	I	88
I	3	I	0.052	I	0.999	I	87
I	.	I	.	I	.	I	.
I	.	I	.	I	.	I	.
I	.	I	.	I	.	I	.
I	21	I	0.358	I	0.934	I	69
I	22	I	0.375	I	0.927	I	68
I COS I SIN I DEGRES I							

```

program trigo(output);
{table sinus/cosinus, en degrés}
var D:integer; {angle en degrés}
    R:real;     {angle en radians}
    i:integer;
begin
    {titre haut}
    for i:=1 to 30 do write ('~'); writeln;
    writeln('1 DEGRES I', 'SIN':7, 'I':3, 'COS':7, 'I':3);
    for i:=1 to 39 do write ('~'); writeln;
    {table}
    for D:=0 to 22 do begin
        R:=3.14159265*D/180.0;
        writeln('I', D:5, 'I':4, sin(R):7:3, 'I':3,
            cos(R):7:3, 'I':3, 90~D:5, 'I':4)
    end;
    {titre bas}
    for i:=1 to 39 do write ('~'); writeln;
    writeln('I':10, 'COS':7, 'I':3, 'SIN':7, 'I':3,
        ' DEGRES I');
    write(' ':10); for i:=10 to 39 do write ('~'); writeln
end.

```

(degré : درجة ؛ angle : زاوية ؛ titre haut : العنوان العلوي ؛ titre bas : العنوان السفلي)

مثال : فرز بمبادلات متتالية .

إن فرز متسلسلة من n عنصر بالترتيب التزايدى يقوم على ترتيبها بشكل أن كل عنصر يكون أصغر ، أو يساوي ، من كل العناصر التي تليه : $T_i \leq T_j \quad \forall i > j$.

هناك فكرة سهلة تقوم على العمل بالتكرار (أو الثنية إلى الوراء) .

- إذا وضعنا في الموقع الأول من المتسلسلة ذات الـ i عنصر ، العنصر الأصغر ، نكون أمام فرز من $i - 1$ عنصر ؛
 - إن متسلسلة من عنصر واحد تُعدُّ مفروزة .

```

program tri(input,output);
{ فرز بالمبادلات المتتالية }
const n=5; { عدد العناصر المطلوب فرزها }
var T:array [1..n] of integer; { عناصر للفرز }
    i,j,k,L:1..n;
    aux:integer;
begin
    { إقرأ المتسلسلة الأولية }
    for k:=1 to n do read(T[k]);

    { الحلقة 1 : ضع في T[1] T[1] الأدنى من T[1...n] }

```

```

for i:=1 to n-1 do
  { الحلقة j : قارن T[i] بكل T[i+1...n] }
  for j:=i+1 to n do
    if T[i]>T[j] then begin      { إذن بادل }
      aux:=T[i]; T[i]:=T[j]; T[j]:=aux end;
  { اكتب المتسلسلة المرتبة }
for L:=1 to n do write(' ',T[L]); writeln
end.

```

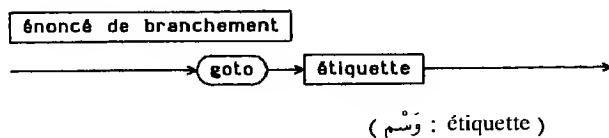
مثال :

	4	9	2	0	3
i = 1	2	9	4	0	3
	0	9	4	2	3
i = 2		4	9	2	3
		2	9	4	3
i = 3			4	9	3
			3	9	4
i = 4				4	9
	0	2	3	4	9

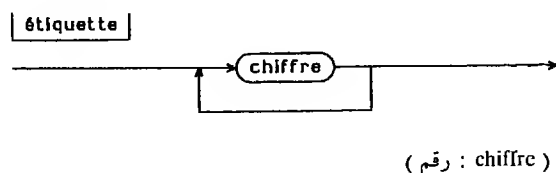
إذن

إن مدة المعالجة (لعدد كبير n) متناسبة مع عدد المقارنات ؛ هذا الفرز هو « فرز بـ n^2 » إنها إحدى الطرق الأسهل للبرمجة ، لكنها إحدى الأبطأ في التنفيذ .

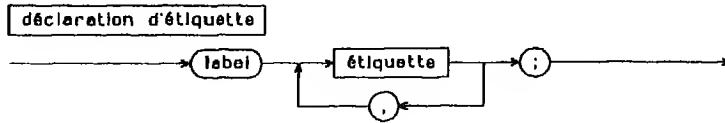
4.3.3 - عبارة الطفور (التفرغ) (Enoncé de branchement)



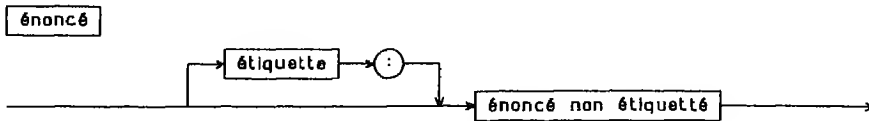
تعني عبارة الطفور أن التنفيذ يجب أن يتابع من النقطة المشار إليها بالوسم في البرنامج . (إنها يمكن أن تسبب الإنهاء الإجباري لعدة إجراءات أو دوال منشطة) .



يتم تمييز الوسومات بواسطة قيمتهم ، في الفترة (0)....9999 (4 أرقام) . يجب التصريح عن كل وسم :



وكذلك تعريفه (بطريقة واحدة) :



(énoncé non étiquetté : عبارة غير موسومة) .

مثال : تردد (fréquence)

أحص ظهور الأحرف في نص لا يحتوي إلا على أحرف كبيرة وتباعدات . في حال ظهور سمة أخرى ، يعد ذلك غلطاً .

ملاحظة : في كل لعب سمات ، تكون الأحرف من 'A' إلى 'Z' مرتبة لكن ليس بالضرورة متتالية (succ ('A') ≤ 'B') ؛ لا يعمل هذا البرنامج بشكل كامل إلا على حاسب آلي تكون الأحرف فيه مكوّدة بطريقة متتالية (لعب سمات ASCII ، BCD ، DISPLAY ...) .

```

program frequency(input,output);
{ احص ظهور الأحرف في نص يحتوي فقط على أحرف كبيرة وتباعدات .
  في لعب السمات يجب أن تكون الأحرف متتالية }
label 1; { للتوقف في حال وجود غلط في المعطيات }
var F:array ['A'..'Z'] of integer; { تردد }
      i,j:'A'..'Z';      c:char; { سمة مقروءة }
begin
  for i:='A' to 'Z' do F[i]:=0;
  while not eof do begin
    read(c);
    if (c<>' ' and ((c<'A' or (c>'Z')) then begin
      writeln('erreur - caractere lu: <',c,>');
      goto 1
    end;
    F[c]:=F[c]+1
  end;
  for j:='A' to 'Z' do writeln(j,F[j]:6);
1: end.

```

(erreur : غلط ؛ caractere lu : سمة مقروءة)

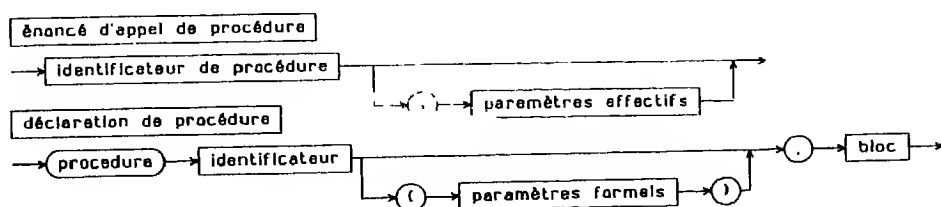
إنّبه : إن استعمال عبارات الطفور تجعل وبسرعة أيّ برنامج ، صعب القراءة وتُضِرُّ بحُسن سير عمله سواء كان ذلك في لغة الباسكال أو في أية لغة أخرى .

4.3 - إجراء (Procédure)

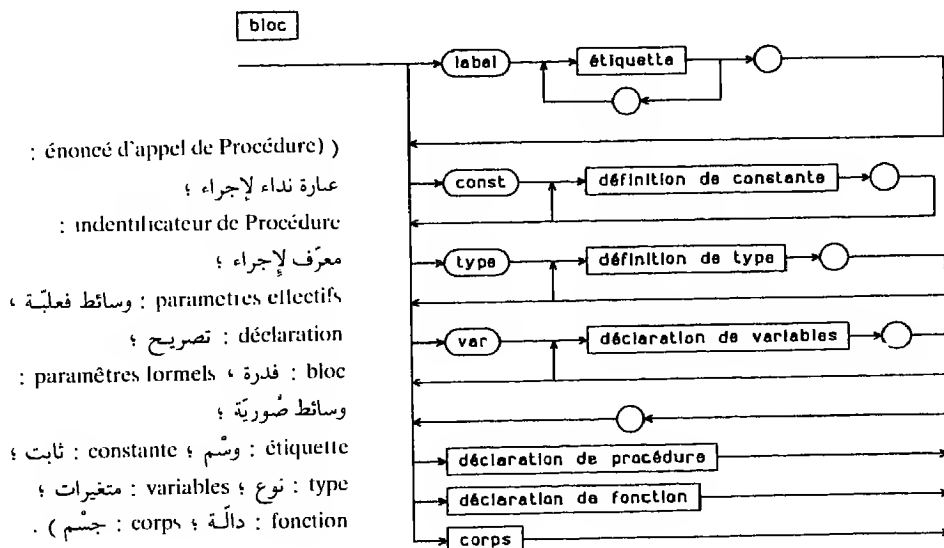
يمكن إعطاء إسماً (معرّف) لعبارة ، والرجوع إليها بواسطة هذا الاسم ؛ تصبح العبارة إجراء ، التصريح عنها تصريحاً لإجراء ، ونرجع إليه بواسطة عبارة نداء لإجراء .

يمكن لتصريح كهذا أن يحتوي على تصريحات وتعريفات (لمتغيرات ، لأنواع ، ...) ، وكذلك على تصريحات أخرى لإجراءات ؛ هكذا فإن الأدوات المعرّفة لا يمكن استعمالها إلا في داخل الإجراء نفسه وتسمى موضعية في الإجراء . ليس لمعرّفيهم أي معنى إلا في الفكرة التي يشكلها تصريح الإجراء .

غالباً ما نستعمل إجراءات ، إما لتسمية عبارة تُستعمل مرات عديدة ، وإما لتحسين قراءة البرنامج عن طريق تحويل أقسام ثانوية من التحليل إلى إجراءات . يتعلّق الإجراء بفعلٍ محدّد ومنفرد .



تذكير :



1.4.3 - كُشِف

نبحث في حسابات بنك عن الأرصدة المدينة غير المغطاة بتسهيلات مصرفية مسموح بها (مأذونة) .

تكون المعطيات على الشكل :

الإذن	وَضْعُهُ	رقم الحساب
(حقيقي)	(حقيقي)	(4 أرقام)

تُعالج فقط الأرصدة التي تنتهي أرقام حساباتها بصفر ؛ تنتهي المعطيات بحساب رقمه 0000 والذي هو خارج إطار المعالجة .
تكون العملية التكرارية إذن من النوع توقّف مع إقْتصار والذي يُترجم بمتاليه :

<i>lire les donnees du premier compte</i>	اقرأ معطيات أول حساب
while <i>compte < > dernier do begin</i>	do begin while الحساب < > آخر
<i>traiter le compte, en comparant position</i>	عالج الحساب بمقارنة
<i>et autorisation</i>	الوضع والإذن
<i>lire les données du prochain</i>	اقرأ معطيات الحساب التالي
<i>compte à traiter</i>	المطلوب معالجته
end	

إن الفعل « اقرأ معطيات حساب » يظهر مرتين ، سنعمل منه إجراءً . المطلوب رؤية كل أرقام الحسابات المتتالية حتى العثور على واحد ينتهي بصفر ؛ إن هذا هو إذن عملية تكرارية من النوع توقّف مع تضمين والذي يُترجم بـ :

repeat <i>lire compte, position, autorisation</i>	repeat اقرأ حساب ، وضع ، إذن
until <i>le dernier chiffre du compte est un zéro</i>	until آخر رقم من الحساب هو صفراً .

سيتم كذلك وَصَف الفعل « عالج حساباً » بواسطة إجراءً وذلك بهدف تأمين قراءة جيّدة للبرنامج ، أخيراً يجب الإشارة إلى الكشوفات أي الحسابات التي يكون الإذن فيها (≥ 0) أقل من الرصيد المدين (< 0) :

if *position < 0 then*
if *abs (position) > autorisation then signaler*

(position : وضع ؛ autorisation : إذن ؛ signaler : أنشُر إلى)

```

program decouvert(input,output);
const dernier=0;
var compte:integer;      { رقم الحساب }
    position,autorisation:real;

procedure lire;
{ ابحت عن الحساب التالي المطلوب فحصه }
begin
    repeat read(compte,position,autorisation)
    until(compte mod 10)=0
end ;

procedure traiter;
{ أشر إلى الحساب المكشوف }
begin
    if position<0.0 then
        if abs(position)>autorisation then
            writeln('compte',compte:7,' decouvert=',
                    -position-autorisation:13:2)
end;

begin
    lire;
    while compte<>dernier do begin
        traiter;
        lire
    end
end.

```

(déouvert : كُشف ؛ compte : حساب ؛ traiter : عالِج ؛ lire : إقرأ) .

مع المعطيات :	الإذن	وضعه	رقم الحساب
	1324	12.00	0.00
	9710	270.50	100.00
	0020	- 986.00	700.00
	0971	- 200.03	100.00
	3640	- 0.27	0.00
	0000	0.00	0.00
			286.00
			0.27

نحصل على : الحساب رقم 20 مكشوف بـ
الحساب رقم 3640 مكشوف بـ

2.4.3 - موضعيّ / إجمالي (Local / global)

لا يمكن إستعمال معرفاً إلاّ

في القدرة التي تمّ التصريح عنه داخلها : إنه موضعي في هذه القدرة .
في القدرات المتراكبة في قدرة التصريح : إنه إجمالي في هذه القدرات .

هكذا ، مع :

```

procedure P1;
type i ...
var j ...
    procedure P2;
    var k ...
        procedure P3;
        var L ...
        corps de P3
    corps de P2
    procedure P4;
    var m ...
    corps de P4
corps de P1
    
```

(**procédure** : إجراء ؛ **type** : نوع ، **var** : متغير ؛ **corps** : جسم) .

- في جسم P1 يمكن استعمال المتغير z ، الإجراءات P2 و P4 (معرفين موضعيين) ، لكن ليس L ، P3 ، K . P1 (إجمالي) يمكن استعماله .

- في جسم P2 ، يمكن استعمال P1 ، z و P2 (معرفين إجماليين) وكذلك K و P3 (موضعيين) .

- في قسم التصريحات من P2 يمكن استعمال النوع i (إجمالي) .

- في P3 ، L هو موضعي ؛ P1 ، i ، j ، P2 ، K و P3 هم إجماليين .

- في P4 ، m هو موضعي ؛ P1 ، i ، j ، P2 ، P4 هم إجماليين ؛ L (من P3) غير ممكن استعماله .

في الإجراء P4 ، المَعْرِفُ l هو بالكامل غير معروف ؛ يمكن إذن التصريح عنه :

```

procedure P4;
var m, L ...
    
```

فإذن المتغير L من P4 ليس له أية صلة مع المتغير L من P3 .

إذا كان قد تم التصريح عن مَعْرِفٍ في فِدرَة ، فإنه من الممكن إعادة التصريح عنه في كل فِدرَة متراكبة في الأولى ؛ ويكون بذلك للمَعْرِفِ مدلول آخر :

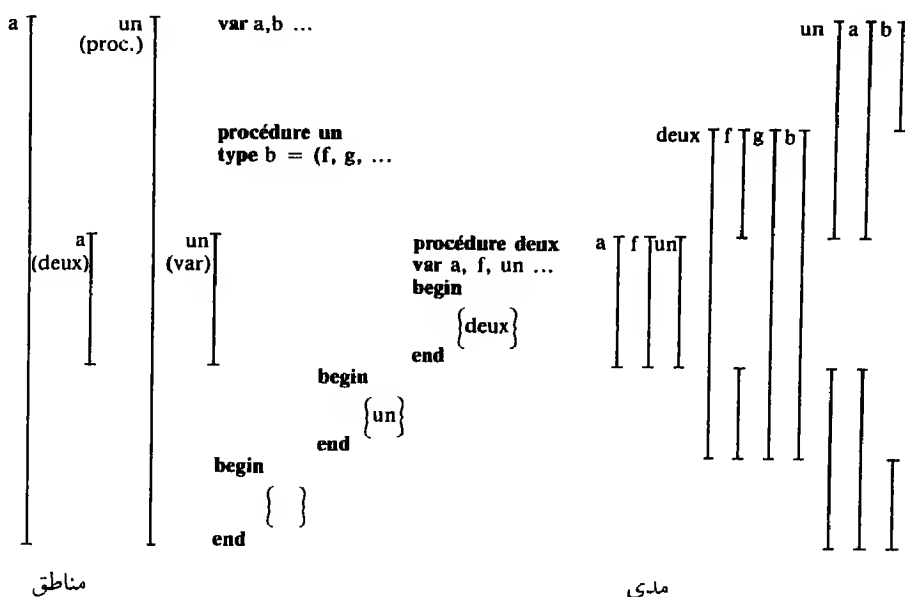
```

...
const a=...
type b=(c,d,e)...
var f,g,h:char...
    procedure P;
    const g=...
    type c=(e,a)...
    var f:char...
    
```

مثال : في جسم P ، $f: = 'x'$ لا تؤثر بالمتغير f الإجمالي ؛ بينما $h: = 'x'$ تؤثر بالمتغير h الإجمالي .

3.4.3 - مدى (Portée)

إن مدى تصريح ، أو تعريف هو جزء من البرنامج حيث يكون التصريح صالحاً ، بمعنى آخر القسم الذي يمكن فيه إستعمال المعرف مع كل خواصه المصرحة . يكون المدى جزء (أو الكل) من منطقتة : إنه منطقته مطروح منها مناطق الأسماء المشتركة المحتملة .
- إن منطقة المعرف لمتغير ، لنوع ، لثابت ، لإجراء أو لدالة هي فدرية (أقسام التصريحات + قسم العبارات) تصريجه أو تعريفه (تحتوي الفدرية ، الفدرات المتراكبة) .
- لنفترض وجود معرف له المنطقة A ؛ إذا كان هناك معرف له نفس كتابة الكلمات وعنده المنطقة B الموجودة ضمن A ، فإذن تكون المنطقة B وكل المناطق التي تحويها خارجة عن إطار مدى المعرف الأولي :



(حالات خاصة : قائمة الوسائط الصورية ، حقل الفقرة) .

قواعد :

- يجب التصريح أو التعريف (أو التعريف مسبقاً) عن كل معرف ؛
- لا يمكن وجود عدة معرفين لهم نفس كتابة الكلمات ، مصرح عنهم أو معرفين في نفس المنطقة ؛

- لا يمكن إستعمال المعرّف إلا ضمن نطاق مداه ؛
- يجب على تصريح أو تعريف المعرّف أن يسبق كل استعمالاته (ما عدا النوع دليل (pointeur) (x) ، مصرّحة type x = ... لها كمنطقة : فدرّة ، غير أنّ type y : x ; x = هي غير سليمة) .

مثال :

```

program portee(output);
var i,j,k,L:integer;
  procedure P1;
    var i,j:integer;
      procedure P2;
        var i,k:integer;
        begin i:=6; k:=6;
          writeln('P2':5,i:5,j:5,k:5,L:5);
          i:=2; j:=2; k:=2; L:=2
        end;
      begin i:=5; j:=5;
        writeln('P1':5,i:5,j:5,k:5,L:5);
        P2;
        writeln('P1':5,i:5,j:5,k:5,L:5);
        i:=1; j:=1; k:=1; L:=1
      end;
begin
  i:=0; j:=0; k:=0; L:=0;
  writeln('prog':5,i:5,j:5,k:5,L:5);
  P1;
  writeln('prog':5,i:5,j:5,k:5,L:5);
end.

```

يعطي النتائج :

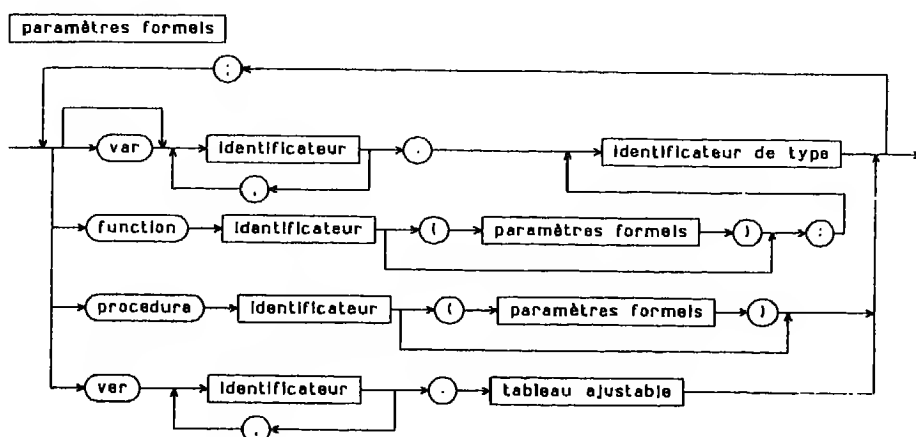
0	0	0	0	
P1	5	5	0	0
P2	6	5	6	0
P1	5	2	0	2
0	0	1	1	

4.4.3 - وسائط (Paramètres)

يمكن أن يكون للإجراء وسائط بعدد محدّد معبّر عنهم في الإجراء بواسطة معرّف : الوسيط الصّوري (paramètre formel) . عند نداء الإجراء ، يجب تحديد الوسيط الفعلي ، (paramètre effectif) ، الذي يحلّ في الإجراء مكان الوسيط الصّوري طيلة فترة تنشيط الإجراء .

إن الطريقة التي بها يحلّ الوسيط الفعلي مكان الوسيط الصّوري هي صيغة إنتقاله :

- وسيط قيمة : كل إحالة الى الوسيط الصوري هي إحالة الى قيمة الوسيط الفعلي (هذا تعبير) .
- وسيط متغير (الكلمة الدلالية Var) : كل إحالة الى الوسيط الصوري هي إحالة الى الوسيط الفعلي ، الذي يجب أن يكون متغيراً .
- وسيط إجراء : يكون الوسيط إسماً لإجراء (أنظر 1.4)
- وسيط دالة : يكون الوسيط إسماً لدالة (أنظر 1.4) .
- وسيط جدول ضبط : مُنْتَقَل بواسطة القيمة أو المتغير ، إنه جدول لا نعرف عدد عناصره لحظة كتابة البرنامج (أنظر 1.4) .



(paramètre formel : وسيط صوري ؛ identificateur : معرف ؛ procédure : إجراء ؛ tableau ajustable : جدول ضبط) .

- وسيط قيمة : يجب أن لا يشتمل نوع الوسيط الصوري على سجل . إن الوسيط الفعلي هو تعبير من نوع متساوق بالنسبة للتعين مع الوسيط الصوري . إنه وسيط معطية : نقل معطية (قيمة تعبير) إلى الإجراء .
- وسيط متغير : يكون الوسيط الفعلي متغيراً له نفس نوع الوسيط الصوري . إنه وسيط معطية ونتيجة : إذا كان للمتغير قيمة ، فإن هذه القيمة صالحة للاستعمال في الإجراء ؛ إذا عيّن الإجراء قيمة للوسيط ، فإنه يتم نقل هذه القيمة إلى المتغير . يتم تحديد الوسيط قبل تنفيذ الإجراء (التيل : دليل ، حقل ، مُتَمَم) .

```

program parametres;
var i,j:char;
procedure P(k:char;var L:char);
begin
  writeln (^P^,k,L);
  k:=succ(k); L:=succ(L)
end;
procedure Q(k:char;var L:char);
begin
  writeln (^Q^,k,L);
  P(k,L);
  writeln (^Q^,k,L)
end;
begin
  i:=^0^; j:=^0^;
  writeln(i,j);
  Q(i,j);
  writeln(i,j);
  Q(j,i);
  writeln(i,j)
end.

```

يعطي النتائج :

00	_____	Q (i, j)
Q00	_____	P (k, l)
P00		
Q01		
01	_____	Q (j, i)
Q10	_____	P (k, l)
P10		
Q11		
11		

ملاحظة : يمكن أن يكون الوسيط الفعلي المتغير ، سجلاً .
 انتبه : يجب أن تتطابق الوسائط الصورية والفعلية من حيث العدد والنوع .
 مثال : إجراءً يحسب مجموع عددين :

```

program P1(input,output);
var a,b,c:real;
procedure somme(a,b:real; var c:real); {c=a+b}
begin c:=a+b end;
begin read(a,b);
  somme(a,b,c); writeln(a,^+^,b,^=^,c);
  somme(a,1.0,c); writeln(a,^+^,1.0,^=^,c);
end.

```

(somme : مجموع)

مثال : إجراءً يحسب مجموع متجهين من 10 عناصر

```
program P2(input,output);
const n=10;
type indice=1..n;
      vecteur=array [indice] of real;
var a,b,c:vecteur;
    i:indice;
    procedure somme(a,b:vecteur; var c:vecteur);
      var i:indice;
      begin for i:=1 to n do c[i]:=a[i]+b[i] end;
    begin
      for i:=1 to n do read(a[i]);
      for i:=1 to n do read(b[i]);
      somme (a,b,c);
      for i:=1 to n do writeln(c[i])
    end.
```

(vecteur : متجه)

مثال : إجراء للفرز

إن الجدول المطلوب فرزه ، معطية من الإجراء ، هو كذلك نتيجته : نصّرح عنه كوسيط متغير :

```
type tableau=array [1..n] of ...
procedure tri(var T:tableau) ...
```

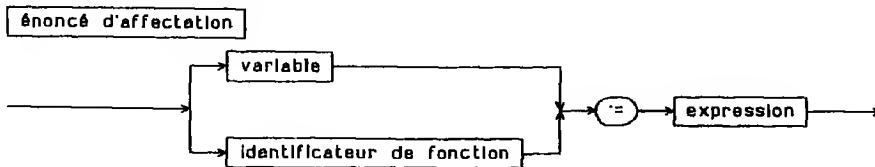
5.3 - دوال (Fonctions)

كالإجراء فإن للدالة عنوان ، فدره ، جسم ، وسائط ، لكن
- تنقل الدالة نتيجة ، من نوع بسيط ،
(مثال $f(a, b: integer) : integer$ function تنقل نتيجة صحيحة) .

- يتم النداء داخل تعبير (أنظر 1.4.2)

(مثال : $x := \dots * f(12, -13) \dots$)

يجب أن تحتوي فدره الدالة على عبارة تعيين واحد على الأقل متعلق بالنتيجة :



(مثال : $f := (a - b) * (a + b)$)

(énoncé d'affectation : عبارة تعيين ; expression : تعبير ; identificateur : معرف ; variable : متغير)

مثال : دالة تحسب مجموع عددين :

```

progra . Fl(input,output);
var a,b:real;
  function somme(x,y:real):real;
    begin somme:=x+y end;
begin read(a,b);
  writeln(a,'+',b,'=',somme(a,b));
  writeln(a,'+',1.0,'=',somme(a,1.0));
end.

```

1.5.3 - مدى (portée)

داخل عنوان إجراء أو دالة ، تكون منطقة معرفي الوسائط هي فدر الإجراء أو الدالة ؛ يكون إسم الإجراء أو الدالة كما معرفي النوع ، داخل الفدر الشاملة . تسمى هذه القواعد قواعد المدى المنظفة .

مثال :

فدر شاملة b)

$$\text{function } F (\boxed{A} : T1; \text{var } \boxed{B} : T2) : T3;$$

فدر الدالة

هكذا يمكن إعادة التصريح عن T1 في الدالة : كما استطاعت قبلاً الفدر الشاملة استعمال A .

2.5.3 - مفعول الحافة (Effets de bord)

نتكلم عن مفعول الحافة عندما تُغيّر دالة أو إجراء ، متغيراً إجمالياً ونعتبر بشكل عام بأن ذلك أضعف من حسن قراءة ودقة البرامج ؛ لكي نخفف من هذه العواقب ، نكتب في ملاحظات كل إجراء ، المتغيرات الإجمالية القابلة للتغيير ، ونمنع كل مفعول حافة عن الدالة .

أ - مثال « جيّد » :

```

var compte:integer;
...
procedure P; { يؤثر بالمتغير الإجمالي compte (حساب) }
begin compte:=compte+1; ... end;

```

(يصلح المتغير الإجمالي لعدّ تنشيطات الإجراء P) .

ب - مثال « سيء »

```
var i,j,k:integer;  a:array [0..99] of integer;
...
function f(a,b:integer):integer;
begin f:=(a+b) mod 100; i:=i+1 end;
...
```

(يصلح «i» لعدّ تنشيطات الدالة) .

تؤدي العبارة $a[f(i,j)] := f(i,k) + i$ إلى نتيجة غير محددة : لا نعرف إذا كان يتم تقييم القسم الأيسر من عملية تعيين قبل القسم الأيمن ، أو العكس (يختلف ذلك من حاسوب إلى آخر وليس بالضرورة محددًا : لا شيء يمنع بأن يكون هذا هو محض صدفة في كل تعيين) .

إذا $i = 2$ و $j = 7$ و $k = 11$

- في الحالة الأولى ، نحصل على :

إلى اليسار : $f(i,j) = 9$ و $i = 3$

إلى اليمين : $f(i,k) = 14$ و $i = 4$

$f(i,k) + i = 18$

يعني $a[9] := 18$

- في الحالة الثانية ، نحصل على :

إلى اليمين : $f(i,k) = 13$ و $i = 3$

$f(i,k) + i = 16$

إلى اليسار : $f(i,j) = 10$ و $i = 4$

يعني $a[10] := 16$

(يمكن كذلك الحصول على

$f(i,k) = 13$ و $i = 3$

$f(i,j) = 10$ و $i = 4$

$f(i,k) + i = 17$

يعني $a[10] = 17$)

6.3 - تمارين

1 - لنفترض معطي تاريخًا ، على شكل ثلاث قيم صحيحة M, J و A . جدّ تاريخ اليوم

التالي (إنته للسنوات الكبيسة) .

2 - في نص مؤلف من أحرف كبيرة ، تباعدات ، نهايات أسطر وسمات أخرى ، نوّد عدّ

- كل زوج من الأحرف ؛ سيتم تجاهل كل السمات التي ليست هي أحرف كبيرة .
- 3 - في فحص يشتمل على P إمتحانات ، N مرشح حصلوا على علامات على 20 . لكل إمتحان z يتطابق مُعامل Cj ؛ لا يُقبل المرشح إلا إذا حصل على معدل أقله 12 ؛ إذا حصل على معدل بين 10 و 12 فسيخضع لإمتحان شفهي للإستحقاق .
- إطبوع في كل مرة قائمة أرقام المرشحين المقبولين ، ثم قائمة المرشحين المقبولين لإجراء امتحان شفهي ، ثم قائمة المرشحين المرفوضين ، وذلك بالترتيب التنازلي للمعدلات .
- 4 - مخطط دَرَجِي (histogramme) : وزّع إلى 20 فترة من نفس الحجم ، نتائج عدة قياسات (اعداد حقيقية) ؛ عَدَد القياسات ليس معروفاً مسبقاً .
- 5 - لنفترض معرفة الإحداثيات (x, y, z) لعشرة نقاط في الفضاء (أوقليدي \mathbb{R}^3) . جِدْ النقطتين الأقرب .
- 6 - حَوِّل عدد روماني ($\leq c$) إلى قيمته العشرية .
- 7 - إنطلاقاً من عدد ليرات الوقود الموضوعية في سيارة بهدف إملاء خزانها ، وترقيم الكيلومترات المبين على العدّاد عند ملء خزان السيارة ، إحسب إستهلاك السيارة بين ملتين لخزان السيارة ومعدل الإستهلاك (بالليترات لكل 100 كيلومتر) .
- 8 - جِدْ كل الأعداد الأوليّة الأصغر من 100 .
- 9 - جِزْ عدد صحيح موجب إلى عوامل أوليّة .
- 10 - جِدْ عدداً صحيحاً ، أصغر من 100 ، والذي يساوي مجموع أرقام مُكعّبه .
- 11 - إحسب القيمة e ، قاعدة اللوغاريتمات ($\ln(e) = 1$) ، عن طريق جمع المتسلسلة $\sum_{n=0}^{\infty} \frac{1}{n!}$ (نوقف عملية الجمع فور الحصول على : دقة محسوبة $\frac{1}{n \star n!}$)
- ملاحظة : $n!$ « تعني » عاملي العدد n ، $n! = \prod_{i=1}^n i$ ، $n \gg n!$
- 12 - تم وضع المبلغ x في البنك للحصول على معدل فائدة مركبة t . ماذا يصبح هذا المبلغ بعد مضي سنة ، سنتين ، . . 20 سنة .
- 13 - يبيّن الجدول التالي العلاقة الرسمية (الضرائبية) بين الفرنك الثابت والفرنك المتداول :

معامل لتطبيقه على سعر الإكتساب أو المصاريف	سنة الإكتساب أو الإستهلاك	مُعامل لتطبيقه على سعر الإكتساب أو المصاريف	سنة الإكتساب (التملك) أو الإستهلاك (المصاريف)
3,74	1967	8,18	1950
3,58	1968	6,99	1951
3,36	1969	6,26	1952
3,14	1970	6,33	1953
3,03	1971	6,36	1954
2,85	1972	6,28	1955
2,66	1973	6,16	1956
2,34	1974	6,00	1957
2,09	1975	5,22	1958
1,91	1976	4,92	1959
1,75	1977	4,74	1960
1,60	1978	4,59	1961
1,45	1979	4,38	1962
1,27	1980	4,18	1963
1,12	1981	4,04	1964
1,00	1982	3,94	1965
		3,84	1966

اكتب الإجراءات التي تُردّ على الأسئلة :

- بأي سعر يجب أن نبيع في الـ 1982 مواداً مُشتراة بـ x فرنك (في الماضي) في السنة A ، كي « لا نخسر أموالنا » ؟
- كم كان يكلف في السنة B ، مواداً كُلفت y فرنك في العام 1982 .
- ما هو معدّل التآكل النقدي بين السنتين c و D ($D > C$) ؟
- في أية سنة كان معدّل التآكل الأكبر ؟
- 14 - إختزال كولاتز (collatz) : لنأخذ عدداً صحيحاً ، لنقسمه بـ 2 إذا كان عدداً زوجياً ؛ لنضربه بـ 3 ونزيد عليه 1 إذا كان عدداً مفرداً ؛ إذا أعدنا الكرة مرات عديدة فإننا سننتهي دائماً بالحصول على 1 . ما هو عدد الإختزالات المطلوبة لكل من الأعداد من 2 إلى 100 ؟ (مثال : 5 : تتطلب 5 إختزالات : 1, 2, 4, 8, 16, 5) .
- 15 - لعبة الحياة : في الإطار $N \times N$ المحدّد لـ N^2 خلية ، نضع في البداية بعض الشاغلين . كل خلية (i, j) تجاور 8 أخرى ووحدها $occ(i, j)$ هي مشغولة .
نتنقل من جيل إلى آخر عن طريق تطبيق القواعد :
1 - يعيش شاغلاً لـ (i, j) إذا $occ(i, j)$ تساوي 2 أو 3 ،

- 2 - يموت شاغل (i, j) إذا كانت $occ_{i,j}$ أقل من 2 (عدد قليل من الناس) أو أكبر من 3 (عدد كبير من الناس) ؛
- 3 - تستقبل الخلية الفارغة (i, j) شاغلاً في حال $occ_{i,j} = 3$
- إنطلاقاً من مجموعة سكانية أوليّة ، قُم بمظاهرة 12 جيلاً متتالياً .

الفصل الرابع

مفاهيم أكثر تقدماً

سيتم في هذا الفصل توضيح وإتمام المفاهيم الأساسية ، للوصول إلى وصف كامل للغة الباسكال . لكي يتم فهم هذا الفصل بشكل أفضل ، فإن معرفة جيدة للتقنيات الأساسية للباسكال تبدو ضرورية .

لقد تمّ تجميع مختلف النقاط المعالجة تبعاً لمواضيعها : إن قراءة هذا القسم من الكتاب ليست بالضرورة متتالية .

سنحاول ، من خلال الأمثلة المقدمة عامة على برامج كاملة ، تبين كيف يتم تكوين برنامج مركّب (مهيكل) ، وكيف أن التفكير بالمعطيات والخوارزمات على مستوى مجرد كفاية ، يقود عمليات البحث المتابعة .

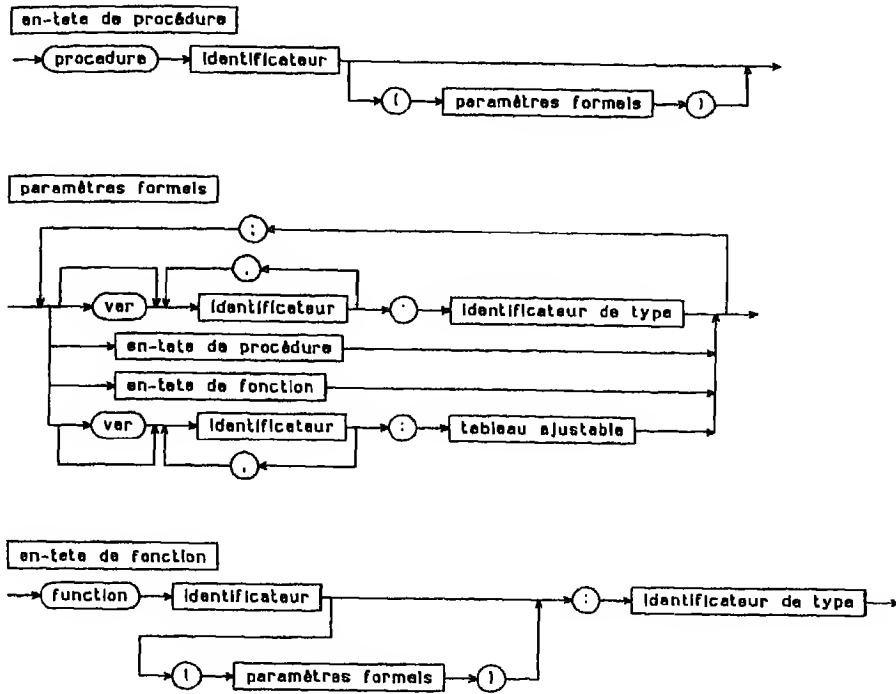
1.4 - وسائط (paramètres)

يمكن أن تكون الوسائط المنقولة إلى إجراء أو دالة ، من إحدى هذه الصيغ :

- وسيط قيمة : الوسيط الفعلي هو قيمة لتعبير (أنظر 4.4.3) ؛
- وسيط متغير (Var) : الوسيط الفعلي هو متغير ، الذي يُمكن للإجراء تعديل قيمته (أنظر 4.4.3) .
- وسيط إجراء (Procedure) : الوسيط الفعلي هو إسم إجراء ؛ ننقل بذلك فعلاً وليس معطية .
- وسيط دالة (function) : الوسيط الفعلي هو إسم دالة ؛ ننقل بذلك مؤثراً وليس معطية .
- جدول ضبيط (1) ، بالقيمة أو بالمتغير : يعرض إذ ذاك الوسيط الصوري نموذجاً لجدول

(1) تُعرّف النظم AFNOR مستويين من اللغات ، 0 و 1 ؛ يحتوي المستوى 1 إضافة إلى المستوى 0 الوسائط جدول ضبيط . لكن كثير من قواعد الباسكال تماثي مع المستوى 0 ، ولا تعالج إذن الجداول الضبيطة .

يُضبط على الجدول المعطى كوسيط فعلي ؛ يمكن بذلك كتابة إجراء أو دالة يعمل على جدول دون المعرفة المسبقة لحدود دلائله .



('en-tête' : عنوان ؛ procédure : إجراء ؛ paramètres formels : وسائط صورة ؛ identificateur : معرف ؛ tableau ajustable : جدول ضبط ؛ fonction : دالة ؛ type : نوع)

1.1.4 - وسيط إجراء وسيط دالة (Paramètre Procédure, Paramètre fonction)
يجب أن يكون الوسيط الفعلي معرفاً لإجراء ، أو لدالة ، جرى تعريفه في البرنامج . بالطبع يجب أن تتطابق وسائطه ، إذا وُجِدَتْ ، مع الوسائط المصرحة في قائمة الوسائط الصورية : مع

Procedure P (Procedure a (i : real; Var c: char));

يمكن تسمية P بـ P(B)

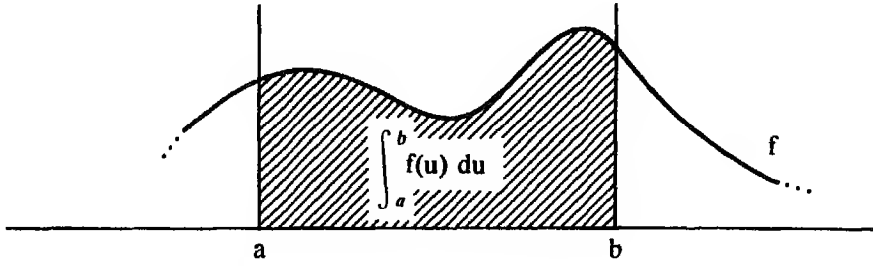
إذا كان للإجراء B العنوان :

Procedure B (x : real ; Var y : char)

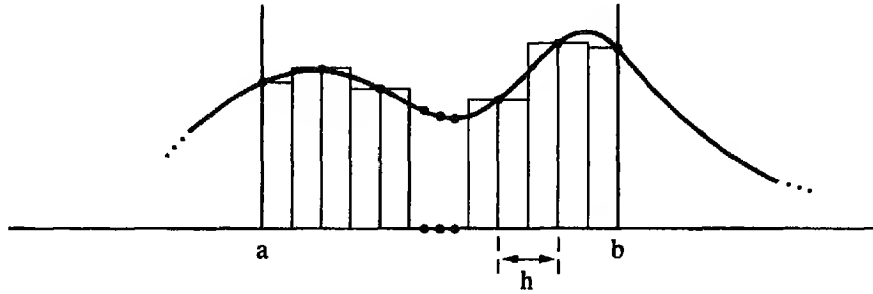
فإذن يمكن في P إستعمال B منقول كوسيط :
مثلاً a (5.2, d)

إن الأساء (i و c) المعطية للوسائط الصورية الخاصة بالإجراء المصرّح a، ليس لها أي مدلول خاص في باقي البرنامج : إنها لا تصلح إلّا للتدليل على وجود الوسائط .
مثال : مُكاملة (intégration)

إن التكامل $\int_a^b f(u) du$ لدالة f هو المساحة المحددة بمحور الإحداثيات الأول ، المنحنى محدّد بـ a و b :



يمكن إجراء الحساب بطريقة تقريبية وذلك بتقطيع المساحة إلى مستطيلات يمكن بسهولة حساب مساحاتها :



إن مساحة مستطيل مركز على محور الإحداثيات الأول x هو إذن $h \star f(x)$
إذا كان يوجد $n - 1$ مستطيل كامل ، ونصفيّ مستطيل (في a و b) ، فإن القيمة التقريبية للتكامل i هي :

$$\int_a^b f(u) du \simeq \left(\sum_{i=1}^{n-1} h \star f(a + i \star h) \right) + h/2 \star f(a) + h/2 \star f(b)$$

$$h = (b - a) / n$$

$$\int_a^b f(u) du \simeq h \star \left((f(a) + f(b))/2 + \sum_{i=1}^{n-1} f(a + i \star h) \right)$$

إذا أخذنا n كبيرة جداً ، فإذا h صغيرة ، نحصل بذلك على قيمة تقريبية للتكامل .

تعطي عملية المكاملة (لدالة حقيقية « جيدة ») نتيجة من نوع بسيط ، حقيقي ؛ يمكن إذن كتابتها على شكل دالة لها كوسائط عددان حقيقيين (الحدود a و b) ، الدالة المطلوب حساب تكاملها (f) ، وعدد الفترات n .

```
function integrale(a,b:real;
                  function f(x:real):real;
                  n:integer)
    :real;
var V:real;
    h:real;      { عرض الفترة : (b-a) / n }
    i:integer;
begin
    V:=0; h:=(b-a)/n;
    for i:=1 to n-1 do V:=V+f(a+i*h);
    integrale:=h*((f(a)+f(b))/2+V)
end;
```

لنفرض أننا نريد حساب تكامل الدالة e^{-u^2} على الفترة $[-1, 1]$ مع 1000 خطوة للمكاملة :

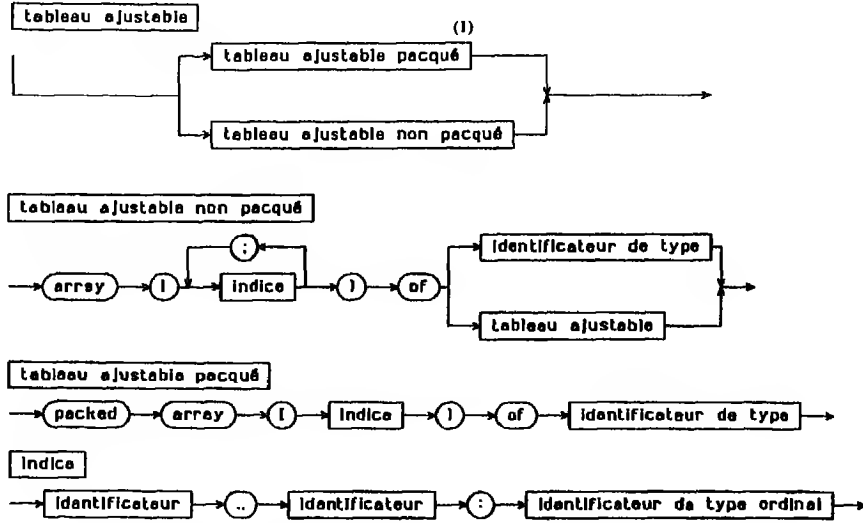
```
program integration(output);
{calcul de "somme de -1 à 1 de e puissance -u2"
 par la méthode des rectangles}

function integrale(
    a,b:real;      { حدود المكاملة }
    function f(x:real):real; { الدالة المطلوب حساب تكاملها }
    n:integer)     { عدد الخطوات }
    :real;
var V:real;
    h:real;      { عرض الفترة : (b-a) / n }
    i:integer;
begin
    V:=0; h:=(b-a)/n;
    for i:=1 to n-1 do V:=V+f(a+i*h);
    integrale:=h*((f(a)+f(b))/2+V)
end;

function cloche (t:real):real;
begin cloche:=exp(-sqr(t)) end;

begin
    writeln('l'integrale de -1 a 1 de e(-t2) vaut',
            integrale(-1,1,cloche,1000):6:3)
end.
```

2.1.4 - وسيط جدول ضبط (Paramètre tableau ajustable)



(tableau ajustable : جدول ضبط ؛ Pacqué : مملب ؛ indice : دليل ؛ identificateur : معرف ؛ Type ordinal : نوع ترتيبي) .

إنتبه : تنتمي الوسائط جداول ضبيطة إلى المستوى 1 من لغة الباسكال ؛ إنه عنصر اللغة الوحيد في هذا المستوى ، بينما تنتمي العناصر الأخرى إلى المستوى 0 . لا يمكن إستعمال الوسيط جدول ضبط إلا إذا كانت نسخة الباسكال المستعملة من المستوى 1 . إن نسخة متطابقة مع المستوى 0 من النظم (norme) لا يمكن بأي حال من الأحوال أن تتضمن إمكانية للجدول الضبيطة .

يمثل معرفي حدود الدليل ، حدود فترة الدليل المطابقة للوسيط الفعلي ؛ يعطي الوسيط الفعلي حدود دلالة إلى الوسيط الصوري . يُستعمل معرفي حدود الدليل في قدرة الإجراء أو الدالة . إن الأدوات المُمثلة بواسطة هؤلاء المعرفين ليست ثوابت ولا متغيرات . يجب أن يتوافق نوع الوسيط الفعلي مع نموذج الجدول الضبيط .

جدول ضبط قيمة : تكون قيمة الوسيط الفعلي (تعبير) منقولة ؛ لا يمكن أن يحتوي الوسيط الفعلي قبلاً على الوسيط جدول ضبط (إلا في حالتين : في وسيط دالة ، في بعض المتغيرات الدليلية) .

جدول ضبط متغير : يمثل الوسيط الصوري الوسيط الفعلي ، الذي يجب أن

(1) سيتم توسيع الشرح عن الرص (Packed) في 1.3.4

يكون متغيراً . لا يمكن أن يكون الوسيط الفعلي مركباً لمتغير معلّـب (Pacquée) .
اختصار : « ؛ » يمكن أن تحل مكان «] of array [» .

ملاحظة : سيتم بحث الأدوات المعلّـبة في 1.3.4 ، يمكن أن يكون الوسيط الفعلي من النوع سلسال (انظر المثال في 2.3.4) ، لكن الوسيط الصوري لا يمكن أن يكون كذلك (ليس معبّر عنه بواسطة النوع جدول) .

مسألة : جداء حسابي (Produit scalaire)

```
procedure ProduitSalaire (
    var A,B:array[inf..sup] of real; var C:real);
var i:integer;
begin
    c:=0.0;
    for i:=inf to sup do C:=C+A[i]*B[i]
end;
```

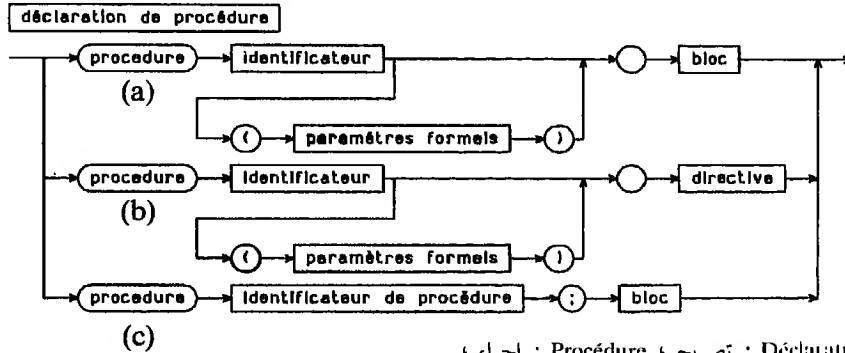
الإستعمال : في برنامجٍ حيث جرى التصريح :

```
type indice = 0..19;
var X,Y:array[indice] of real;
    Z:real;
```

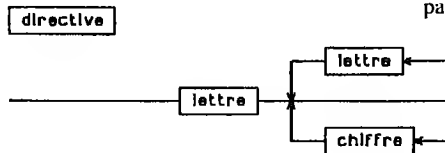
يمكن كتابة : Produit scalaire (x, y, z)

3.1.4 - توجيهات (Directives)

: يمكن أن يظهر التوجيه إلاّ مشتركاً مع عنوانٍ لإجراء أو دالة ؛ إنه يسمح بفصل العنوان عن الفقرة المقابلة : بدون توجيهات ، تتبع الفقرة العنوان :



(Déclaration : تصريح ؛ Procédure : إجراء ؛
identificateur : معرف ؛ bloc : فقرة ؛
paramètres : وسائط صورية ؛ directives : توجيه ؛
formels : حرف ؛ chiffre : رقم) .



في الحالة (ii) يكون العنوان والقدرة متتاليين .
تتوافق الحالات (b) و (c) مع فصلٍ للعنوان والقدرة ؛ هكذا مع التوجيه forward
(أنظر 2.4) ، الذي هو التوجيه الوحيد المطلوب من قبل النظم ، إلى تصريح العنوان
يجب أن ينماشى بعدئذٍ تصريح القدرة وذلك في نفس قسم التصريحات :

(b) **procedure P (i : integer); forward;**

(c) **...
procedure P;
var ...
begin ...
...
end;**

(لا نكرر الوسائط ، المعروفة سابقاً) .

هذا التنويط يسمح بالتكرارية المتقاطعة (récursivité croisée) المشروحة في
(2.4) .

يوجه توجيه آخر في عدد من الحاسوبات : external . إنه يحدد بأن قدرة الإجراء أو
الدالة هي خارج قدرة البرنامج ، أي أن البرنامج الحاوي على التصريح (a) مع
external ، لا يحتوي على التصريح (b) ؛ هذا التصريح (b) قد تم تصريفه على حدة .

تأ أنه يجب التصريح عن كل معرفٍ قبل إستعماله ، فإن استعمال التوجيه يسمح
بتحديد إسم ونوع وسائط إجراء أو دالة ، دون الحاجة الى التحديد المباشر للأفعال .

2.4 - التكرارية (Récursivité)

إن تنشيط إجراء أو دالة يحمي المتغيرات المصرحة في القدرة المرافقة ؛ قبل التنشيط ،
أو بعد « العودة إلى المنادى » ، لا وجود لهذه المتغيرات ، أي لا يمكن بلوغهن .

إذا تمّ من جديد تنشيط قدرة نشطة سابقاً ، فإن المتغيرات المصرحة تُحْيى من
جديد ، وتصبح قيم المتغيرات المرافقة للتنشيط القديم غير قابلة للبلوغ (بالرغم من أن
هذه المتغيرات الجديدة والقديمة مرتبطة بنفس المعرف) ؛ نجد ثانيةً هذه القيم القديمة عند
إزالة تنشيط القدرة الجديدة .

إن التكرارية هي فعلٌ تنشيط قدرة نشطة سابقاً ؛ إنه مفهوم قريب من التثنية إلى
الوراء (récurrence) . مثلاً يُعرّف القاسم الأكبر المشترك PGCD لعددین a و b بالتثنية إلى
الوراء كما يلي :

$\text{pgcd}(a, b) = u_n$, حيث n هو الدليل الأول بشكل
 $\begin{cases} u_i = v_{i-1} \\ v_i = u_{i-1} \bmod v_{i-1} \end{cases} \quad \begin{cases} u_0 = a \\ v_0 = b \end{cases} \quad \text{أن } v_n = 0$

مثال : $u_0 = 25 \quad v_0 = 10$
 $u_1 = 10 \quad v_1 = 5$
 $u_2 = 5 \quad v_2 = 0 \quad u_n = 5$

يُكتب تعريف عملية الـ pgcd إذن :

$\text{pgcd}(u, v) = \bar{u} \text{ si } v = 0, \text{pgcd}(\bar{v}, \bar{u} \bmod \bar{v}) \text{ sinon}$

($\bar{}$: إذا \neq sinon : وإلا)

وفي لغة الباسكال على شكل دالة :

```
program PlusGrandCommunDenominateur(input,output);
var a,b:integer;
function pgcd(u,v:integer):integer;
begin
  if v=0 then pgcd:=u
  else pgcd:=pgcd(v,u mod v)
end;
begin
  read(a,b);
  writeln('pgcd( ',a:1, ', ',b:1, ')= ',pgcd(a,b):1)
end.
```

مع المعطيات $a = 25$ و $b = 10$ ، يكون تنالي الحسابات :

التنشيط الأول : $\text{pgcd}(25, 10)$
 $u = 25$
 $v = 10$
 التنشيط الثاني : $\text{pgcd}(10, 25 \bmod 10)$
 $u = 10 \quad 25$
 $v = 5 \quad 10$
 التنشيط الثالث : $\text{pgcd}(10, 10 \bmod 5)$
 $u = 5 \quad 10 \quad 25$
 $v = 0 \quad 5 \quad 10$

تسدد إذن نتيجة التنشيط الثالث $(\text{pgcd} := u)$ 5

من تحديد يأخذ التنشيط الثاني $u = 10 \quad 25$ ،
 القيمة $v = 5 \quad 10$

5 يساوي $\text{pgcd} := \text{pgcd}(v, u \bmod v)$

التشيط الثالث

من جديد يأخذ التشيط الأول $u = 25$ ، النتيجة تساوي 5 .

$v = 10$

النتيجة : $\text{pgcd}(25, 10) = 5$

إن الكتابة $\text{pgcd}(\dots) :=$ في دالة من نفس الإسم هي نداء تكراري . لكن يمكن
لحساب الـ pgcd أن يُكتب أيضاً على شكل تكرارية بسيطة⁽¹⁾ ؛ في المقابل ، فإن حساب
دالة « أكرمان » (Ackermann) لا يمكن أن يُردّ بشكل منسّق إلى تكرارية :

$$\begin{cases} \text{ack}(0, j) = j + 1 \\ \text{ack}(i, 0) = \text{ack}(i - 1, 1) \\ \text{ack}(i, j) = \text{ack}(i - 1, \text{ack}(i, j - 1)) \end{cases}$$

```
program Ackermann(input,output);
var x,y:integer;  compte:integer;

function ack(i,j:integer):integer;
begin
  compte:=compte+1;
  if i=0 then
    ack:=j+1
  else
    if j=0 then
      ack:=ack(i-1,1)
    else
      ack:=ack(i-1,ack(i,j-1))
    end;
end;
begin
  read(x,y);
  write('ack( ',x:1,' ',y:1,' )=');
  compte:=0;
  writeln(ack(x,y):1,'   en ',compte,' appels')
end.
```

(إن عبر المتغيّر compte هو مفعول حافة للدالة ack) .

إن العيين : $\text{ack} := \text{ack}(i - 1, \text{ack}(i, j - 1))$

يمكن أن يكتب كذلك : $\text{ack} := \text{ack}(i - 1, f(i, j))$

حيث $f(i, j) = \text{ack}(i, j - 1)$

⁽¹⁾ while $v < 0$ do begin $r := u \bmod v$; $u := v$; $v := r$ end;
 $\text{pgcd} := u$;

إذا أردنا التصريح عن هاتين الدالتين في نفس المستوى دون أن نراكيهم ، فإنه لا يمكننا تصريح f ومن ثم ack ، ذلك لأن f تُنادي ack التي لم تُصرِّح حتى الآن ؛ يوجد تكرارية متقاطعة ونستعمل التوجيه forward (أنظر 3.1.4) :

```
function ack (i,j:integer):integer; forward;

function f(i,j:integer):integer;
begin
  f:=ack(i,j-1)
end;

function ack;
begin
  compte:=compte+1;
  if i=0 then
    ack:=j+1
  else
    if j=0 then
      ack:=ack(i-1,1)
    else
      ack:=ack(i-1,f(i,j))
    end;
end;
```

١ هذا ما يسمح باستعمال الدالة f في مكان آخر غير ack) .

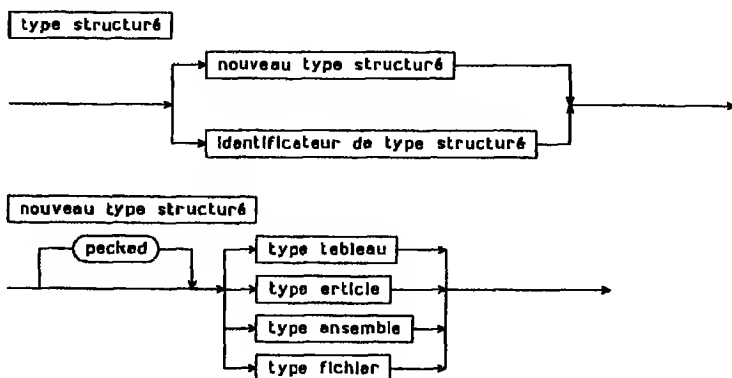
3.4 - الرصّ ، سلاسل السمات (compactage, chaines de caractères)

إن الرصّ الذي ينتج عنه تقليل للأحجام في الذاكرة ، هو ضروري في حالتين :
 - لتوفير من الحجم في الذاكرة لبرنامج يُعالج متغيرات مركّبة ؛ الأداة of char [0...
 - لتوفير من الحجم في الذاكرة لبرنامج يُعالج متغيرات مركّبة ؛ الأداة of char [0...
 array [10000] تحتل 10000 كلمة في كثير من الآلات ؛ عند رصّها ب Packed array فإنها لا تحتل ، تبعاً للدالة ، إلا 5000 ، 2500 [IBM 370) أو
 1000 (CDC Cyber) ؛

- لتسهيل معالجة النصوص ؛ إذا كان قد تم تصريح C of char [1..3] يجب كتابة
 C [1] : = 'R' ؛ C [2] : = 'R' ، C [3] : = 'T' أو (C [3] = 'T') and (C [3] = 'R')
 (C [1] = 'T') and (C [1] = 'R') بينما إذا تم تصريح Packed array [1..3] of char ،
 فإن (C [1] = 'T') and (C [1] = 'R') أو C = 'TRI' .

1.3.4 - الرصّ (Compactage)

إن ظهور الرمز Packed في تعريف من نوع مركّب يعني أن هذا النوع هو مُعلَّب ؛
 إذن حجم القيم في الذاكرة يُخفّض إلى أقل قدر ممكن ، في مقابل خسارة في فعالية
 العمليات على هذه المتغيرات فيما يخص الوقت وكذلك المكان (للبرنامج) .



(type structuré : نوع مركب ؛ nouveau : جديد ؛ identificateur : معرف ؛ tableau : جدول ؛ article : مقرة ؛ ensemble : مجموعة ؛ fichier : سجل) .

في نوع معلّب ، لا يعلّب مركّب الذي يكون هو نفسه مركّب إلا إذا أشير إليه صراحة معلّب .

تبعات الرّص :

- إن نوع جدول معلّب ، ذي دليل منطلق من 1 ، ومحتويًا لسمات ، هو نوع سلسال ذو خصائص محدّدة ؛

- إن وسيطاً فعلياً متغيراً لا يمكن أن يكون مركّباً لمتغير معلّب ؛

- لكن يمكن لوسيط من الإجراء read (إذن readln) أن يكون مركّباً لمتغير معلّب ؛ كذلك فإن وسيطاً من الإجراء write (أو writeln) الذي هو تعبير ، يمكن أن يكون مركّباً لمتغير معلّب .

إن منفذاً إلى مركّب لمتغير معلّب يمكن أن يظهر في كل تعبير ، وفي تعيين ، في القسم الأيسر أو الأيمن . بالمقابل ، فإن نوعاً معلّباً ونوعاً غير معلّب ليسا متساويان بالنسبة للتعيين (إنها ليسا من نفس النوع) ؛ يمكن إذن إستعمال إمّا تكرارية لتعيينهم مركّب بمركّب ، إمّا استعمال الإجراءات المعرفة سلفاً Pack و unpack .

لتفرض التصريح

```

var a: array [s1] of T;
    z: packed array [u...v] of T;

```

- يسمح الإجراء pack بالرّص :

pack (a, i, /) مكافئ لـ :

```
begin k:=i;
  for j:=u to v do begin
    z[j]:=a[k];
    if j<>v then k:=succ(4)
  end
end
```

- يسمح الإجراء **unpack** بانفضاض الرص :
unpack (z, a, i) مكافئ لـ

```
begin k:=i;
  for j:=u to v do begin
    a[k]:=z[j];
    if j<>v then k:=succ(k)
  end
end
```

حيث j و k هما متغيران مساعدان .

2.3.4 - سلاسل (Chaines)

كل نوع **Packed array [1..n] of char** ، حيث تكون $n > 1$ ، هو نوع سلسال من n سمة . إن ثابتاً 'xxx' من n سمة هو كذلك من النوع سلسال من n سمة ؛ يصبح إذن التعيين إلى متغير من نفس النوع ممكناً .
الخصائص :

- يمكن أن يظهر المتغير أو الثابت ، من النوع سلسال في قائمة وسائط write ؛
- نطبق مؤثرات العلاقة .

= <> < <= >= >

على النوع سلسال ، تبعاً للترتيب المعجمي المحدد بلعب السمات ؛ يجب أن تكون القيمتين الموصولتين بالعلاقة متساويتين ، أي لهما نفس عدد السمات . (من المفيد الملاحظة بأن العلاقات لا تطبق على النوع جدول) . تتعلق قيمة تعبير مثل 'ILYA' < c بلعب السمات المستعمل .

لا يمكن لمتغير من النوع سلسال أن يكون وسيطاً لـ read ؛ يجب إستعمال عبارة مثل .

```
for i:=1 to N do read(V[i])
```

حيث تمّ التصريح عن V

```
var V : packed array [1..N] of char
```

إن إجراء عاماً لقراءة السلاسل يكتب :

```

procedure lireChaine (
    var X:packed array[un..max:integer] of char);
label l;
var car:integer;
begin
    for car:=un to max do
        if eof then begin
            writeln('eof lors d'une lecture de chaine')
            goto l
        end
        else
            read(X[car]);
    l:end;

```

إبه من السهل إذن كتابة إجراءات تجري عمليات على سلاسل : تعيين سلاسل
(ذات أطوال مختلفة) ، تنضيد (concaténation) ، إستخلاص (extraction) ، ...

مثال : البحث في جدول من السمات
إكب برنامجاً يسمح بالبحث عن قيمة الإنتاج القومي الخام (PNB) للفرد
(بالدولار) ، لبلد معين .

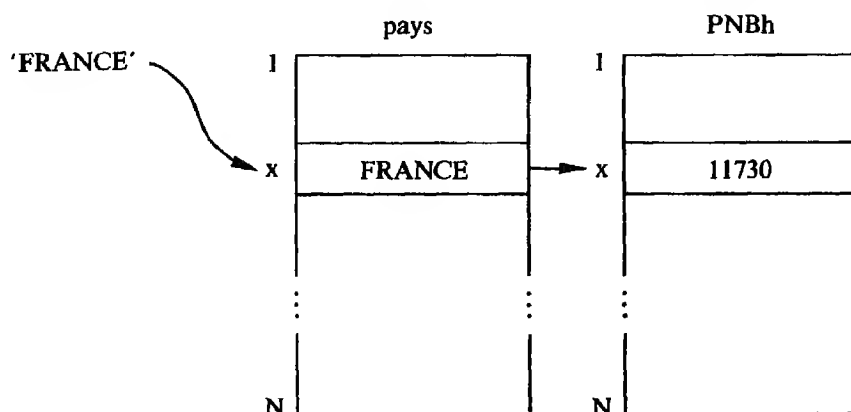
يجب تهيئة جدول مطابقة « بلد ← PNB » للفرد والذي سيتم وصفه في الباسكال
بواسطة المتغيرات :

```

pays: array[1..N] of packed array[1..1] of char;
PNBh: array[1..N] of integer;

```

إن موقع إسم البلد في الجدول الأول ، يعطي في الجدول الثاني قيمة الإنتاج القومي
الخام للفرد فيه (PNBh) :



فأذن نحصل على الصيغة الأولى :

```
1) program recherche(input,output);
    { البحث عن البلد PNBh للفرد لبلد معين }
    const N=?; L=?;
    type chaine=packed array[1..L] of char;
    var pays: array[1..N] of chaine
        PNBh: array[1..N] of integer;
        nom: chaine;
    begin
        initialiser les tableaux pays et PNBh
        repeter
            demander un nom ( اطلب اسماً )
            calculer x son indice dans ( pays في الجدول )
            et écrire PNBh[x] PNBh [ x ]
            en excluant le dernier nom: "fin"
        end.
    { لا يأخذ أحسن الاسم الأخير "fin" }
```

2 - « دَمَث الجداول Pays (بلد) و PNBh » . نقرّر بأنه سيوجد N سطر معطيات ، كل واحد منه مؤلف من عدد صحيح ومن سلسل ؛

```
procedure initialiser;
var ligne : integer;
begin
    for ligne:=1 to N do
        { اقرأ PNBh [ سطر ] و
        Pays [ سطر ] وانتقل إلى السطر التالي }
    end;
```

3 - « اقرأ PNBh [سطر] و Pays [سطر] ، وانتقل إلى السطر التالي » تكتب :

```
read(PNBh[ligne]);
lire pays[ligne] { اقرأ Pays [ سطر ] }
readln
```

4 - اقرأ Pays [سطر] ، يعني قراءة سلسل غير كامل :

```
procedure lireChaine(var c:chaine);
var i,j:integer;
begin
    i:=0;
    while not eoln do begin i:=i+1; read(c[i]) end;
    for j:=i+1 to L do c[j]:=' ';
    readln
end;
```

إذا أهملنا الحالات التي يكون فيها السلسل المطلوب قراءته طويلاً ، أو متوقفاً على نهاية سجل .

5 - «répéter .. en excluant» (كرّر . . . لا تأخذ بالحسبان) تُترجم بواسطة
عبارة while مسبوقه بالحصول على أول عنصر مطلوب معالجته :

```
demandeur un nom ( اطلب إسماً )
while nom<>`fin` do begin
    calculer x son indice dans le tableau pays, et
    écrire PNBh[x] ( احسب x دليله في الجدول Pays ) وإكتب [ x ] PNBh
    demander un nom
end;
```

6 - « اطلب إسماً » هو نداء لإجراء سبق تعريفه :
lire chaine (nom)

7 - « احسب x دليله في الجدول pays ، . . . » يمكن ترجمته بعدة أشكال :
- إذا احتوى الجدول على الأسماء بحالة غير مرتبة ، نجري عملية بحث متتالية :

```
x:=0;
repeat x:=x+1 until (pays[x]=nom) or (x=N);
if pays[x]<>nom then writeln( 'pays inconnu: ',nom)
else writeln( 'PNB/hab( ',nom, ' )= ',PNBh[x] );
```

- إذا كان الجدول مرتباً ، يمكن تسريع عملية التنقيب (dichotomie فرقان)

لقد أتبعنا طريقة التدقيق المتتالي : كل فعلٍ معقد هو مركب من عدة أفعال التي
يمكن كتابتها مباشرة ، إذا كانت بسيطة ، أو وصفها بطريقة عديمة الشكل بواسطة جمل
فرنسية لكي يتم تشرحها فيما بعد .

نجمع البرنامج الكامل كل هذه المراحل :

```
program recherche(input,output);
{ الحظ عن PNB للتردد في بلد معين }
{ القائمة الأولية للبلدان غير مرتبة }
const N=16; { عدد البلدان }
      L=15; { الطول الأقصى للإسم }
      fin=`fin`;
type chaine=packed array [1..L] of char;
var pays:array [1..N] of chaine; { قائمة البلدان }
    PNBh:array [1..N] of integer;
    nom:chaine;
    x:integer; { دليل في الجداول Pays و PNBh }

procedure lireChaine (var c:chaine);
var i,j:integer;
begin
    i:=0;
    while not eoln do begin i:=i+1; read (c[i]) end;
    for j:=i+1 to L do c[j]:=' ';
```

```

    readln
end;

procedure initialiser;
var ligne:integer;
begin
    for ligne:=1 to N do begin
        read(PNBh[ligne]); lireChaine(pays[ligne])
    end
end;

begin initialiser;
    lireChaine(nom);
    while nom<>fin do begin
        x:=0;
        repeat x:=x+1 until (pays[x]=nom) or (x=N);
        if pays[x]<>nom then writeln('pays inconnu: ',nom)
        else writeln('PNB/hab(',nom,',')= ',PNBh[x]);
        lireChaine (nom)
    end
end.

```

11730france	المعطيات الأولية
12180belgique	
13590allemagne	
7920grande-bretagne	
13520suede	
11360etats-unis	
9890japon	
11330ialande	
4880irlande	
16440suisse	
1460turquie	
26080qatar	
270haiti	
1930roumanie	
6710rda	
80bouthan	
suisse	
france	
usa	
etats-unis	
fin	

PNB/hab(suisse)	=16440	: النتائج
PNB/hab(france)	=11730	
pays inconnu: usa		
P.B/hab(etats-unis)	=11360	

عندما يتم فرز جدول أسماء البلدان ، يمكن إجراء التنقيب كما الحال مع القاموس :
نجرّب في الوسط ؛ إذا كان الاسم المطلوب قبل ، نعيد إجراء عملية التنقيب في القسم
الأول ، وإلا في القسم الثاني :

التنقيب في الفترة $a \dots b =$

التنقيب في $\frac{a+b}{2}$ إذا بلد $\left[\frac{a+b}{2} \right] <$ الاسم ، وإلا

التنقيب في $\frac{a+b}{2} ..$

« إحسب x دليله ... » تُكتب إذن :

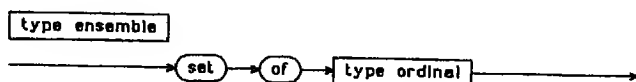
```
u:=1; b:=N+1;
repeat
  c:=(a+b)div2;
  if nom<pays[c] then b:=c else a:=c
until b<=(a+1);
if nom=pays[c] then writeln(PNBh[c])
else writeln('paysinconnu');
```

(Pays inconnu : بلد مجهول)

هذه الطريقة المسماة بالتنقيب الفرقاني تصبح ملائمة لـ $N =$ عدد كبير (عدد المقارنات متناسب مع $\log_2 N$ ، بينما بالنسبة للتنقيب المتتالي يكون تقريباً مساوياً لـ $N/2$).

4.4 - مجموعات (ensembles)

لكي نعرف إذا كانت قيمة متغير من النوع سمة هي التكويد لرقم ، يمكن بالطبع كتابة $(x \geq '0')$ and $(x \leq '9')$ ، لكن من الأسهل إستعمال فكرة المجموعة : $'0' \dots '9'$ $x \in$. تُكوّن المجموعة إنطلاقاً من نوع قاعدي (ترتيبي ، هذا ما يستبعد النوع حقيقي) :



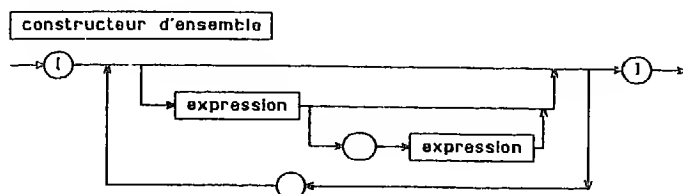
```
A:set of char
b:set of (chene,hetre,bouleau,pin,noyer)
c:set of 0..9
```

مثال :

(ordinal : ترتيبي)

```
b:=[chene,bouleau,pin]
```

يمكن أن تكون قيمة من نوع مجموعة بواسطة مُكوّن مجموعة ، الذي هو عامل لتعبير (انظر 1.3) :






```
if x in ['A', 'E', 'I', 'O', 'U', '0'..'9'] then : مثال
    VovelleOuChiffre
```

(constructeur d'ensemble : مَكُونٌ جموعة ، expression : تعبير)

بالنسبة لكل الأنواع - مجموعات ، تنوّط المجموعة فراغ ب - □

لا يمكن إستعمال القيمة التي يمثلها مكُون مجموعة ، في عمليات أو تعيينات إلا مع مجموعات من نوع متساوق ، يمكن كتابة $A := ['0', '1', '2']$ لكن ليس $A := [0, 1, 2]$:

ملاحظة : إن المجموعة المعرفة في لغة الباسكال هي من الناحية الرياضية مجموعة أجزاء النوع القاعدي . على كثير من الآلات ، تمثل المجموعة بواسطة سلسل من البتة ، هذا ما يفرض قيودا على النوع القاعدي : عدد عناصر محدد ، وقيمة دنيا (صفر) .

+	union	إحاد		$A \cup B$	المتأثرين و النسخة هم من نفس النوع
-	différence	فرق		$A - B$	
☆	intersection	تقاطع		$A \cap B$	
=	égalité	مساواة		$A = B$	المتأثرين هم مجموعات من نفس النوع ، النتيجة هي بولنة
< >	inégalité	مساواة		$A \neq B$	
< =	inclusion	مضمن (واسع)		$A \text{ dans } B : A \leq B$	
> =	inclusion	مضمن (واسع)		$B \text{ dans } A : A \geq B$	
in	appartenance	سواء		$x \in A$	من النوع القاعدي للمجموعة A ، نتيجة بولنة

لا تطبق الإجراءات read و write على المجموعات ، لكن من الممكن كتابتها لكل
نوع مجموعة من الأعداد الصحيحة أو من السمات :

```

type base= 0..31;
    ens= set of base;

procedure writeSet(S:ens);
var x:base; suivant:boolean;
begin
    write('^'); suivant:=false;
    for x:=0 to 31 do
        if x in S then begin
            if suivant then write(', ',x:1) else write(x:1);
            suivant:=true;
        end;
    write(']')
end;

procedure readSet(var S:ens);
var c:chr; x:integer;
begin S:=[];
    read(c);
    if c='[' then begin read (c);
        repeat {lire un nombre}
            x:=0;
            if c in ['0'..'9'] then begin
                while c in ['0'..'9'] do begin
                    x:=x*10+ord(c)-ord('0'); read (c) end;
                if x in [0..31] then S:=S+[x]
                else writeln('erreur: hors intervalle')
            end
        until c=']'
    end
end

```

```

else
  if c<>']' then
    writeln('erreur: caractere illegal');
  if c=',' then read(c)
  else
    if c<>']' then begin
      writeln ('erreur: ] attendu'); c:=']' end
    until c=']'
  end
  else writeln('erreur: [ attendu')
end;

```

(erreur : غلط ، horsintervalle : خارج الفترة ؛ Caractere illégale : سمة محظورة ؛ attendu : متطر)

5.4 - فقرات مع مشتقات ، عبارة مع (Articles avec Variante, Enoncé Avec)

يتم تمثيل أداة ذات عدة مركبات من أنواع مختلفة بواسطة فقرة . مثلاً يمكن أن تكون الدائرة :

record centre : xy; rayon : real end (avec type xy = record
x, y : real end)

(centre : مركز ؛ rayon : شعاع)

وقطعة خطٍ مستقيم :

record origine, extrémité : xy end

(origine : نقطة البدء ؛ extrémité : طرف)

لكن إذا أردنا وصف شكل هندسي ، وَجَبَ تحديد شكل الخط (عادي ، منقوط . .) ولونه (أسود ، غير مرئي . . .) ، وكذلك احداثيات الشكل ؛ إذا كان دائرة فإنَّ احداثيات القطعة غير مفيدة :

خط (trait) :			(عادي ، منقوط)
لون (couleur) :			(أسود ، غير مرئي)
أداة (objet) :			(دائرة ، قطعة ، نقطة)
أداة = نقطة (point)	أداة = قطعة (segment)	أداة = دائرة (cerce)	
احداثيات : xy	نقطة البدء	مركز : xy	
	طرف : x, y	شعاع : real	

إن الحقول التي يعتمد وجودها على المبيّن (indicateur) أداة (objet) ، سيتم كتابتها في لغة الباسكال في مشتقات فقرة :

```

type xy= record x,y:real end;
typeObjet= (cercle,segment,point);
figure=record trsit: . (normal,pointille);
          couleur: (noir,invisible);
          case objet:typeObjet of
            cercle: (centre:xy; rayon:real);
            segment: (origine,extremite:xy);
            point: (coord:xy)
          end;

          case objet:typeObjet of
            cercle:...

```

إن الكتابة

```

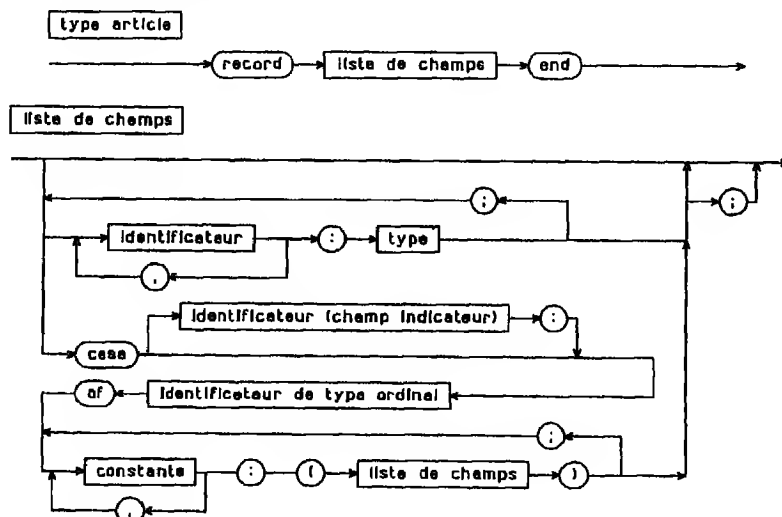
objet:typeObjet;
case typeObjet of
  cercle:...

```

هي اختصار لـ

الحقل objet هو حقل مبيّن ، الذي تسمح قيمته بانتقاء لمشتق . وجب أن تكون لديه قيمة قبل أن يُبلغ حقلاً للمشتق .

إذا ساوى الحقل المبيّن نقطة (point) ، فوحده الحقل coord (إحداثيات) يمكن بلوغه (مركز ، ... ، طرف يمكن بلوغهم) ، هذا ما هو مطابقاً لما أردنا وصفه .



champ : معرف ؛ identificateur : قائمة الحقول ؛ liste de champ : نوع فقرة ؛ type article : حقل مبيّن ؛ type ordinal : نوع ترتيبي ؛ constante : ثابت .

بظهر القسم المشتق في نهاية وصف الفقرة ، بعد القسم الثابت إذا وُجد . قبل بلوغ
 حتماً مُسبق يجب إخبار قيمة الحقل الميَّس ، يمكن أن تكون عبارة « الحالة » مفيدة :

مع `var F:figure` يمكن كتابة

```

case F objet of
  cercle: distance:=abs(sqrt(sqr(F.centre.x)+
                               sqr(F.centre.y))-F.rayon);
  segment: ...
  point: distance:=sqrt(sqr(F.coord.x)+sqr(F.coord.y))
end
    
```

عكس

```

F.trait:=normal;      F.couleur:=noir;
F.objet:=cercle;      F.centre.x:=12.7;   F.centre.y:=-0.9;
F.rayon:=1.0;
    
```

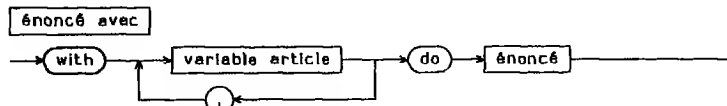
هذا ما يمكن كتابته بطريقة مختصرة بواسطة العبارة مع (Avec) :

```

with F do begin
  trait:=normal; couleur:=noir;
  objet:=cercle; rayon:=1.0;
  with centre do begin x:=12.7; y:=-0.9 end end
    
```

(distance : مسافة ، coord : إحداثيات)

إن بلوغ المتغير فقره وفقاً لـ `with` تتم قبل تنفيذ العبارة وفقاً لـ `do` وهذا البلوغ يوجد
 إسناداً للمتغير طيلة مدة تنفيذ العبارة .



`with v1,v2,..vn do E`

هي مكافئة لـ :

```

with v1 do
  with v2 do
    with vn do
      E
    
```

لكن `with A[i] do begin i:=i+1; x:=... end`

هي مختلفة عن `begin i:=i+1; A[i].x:=...`

(`énoncé avec` : عباره مع ' variable article ' متغير فقره)

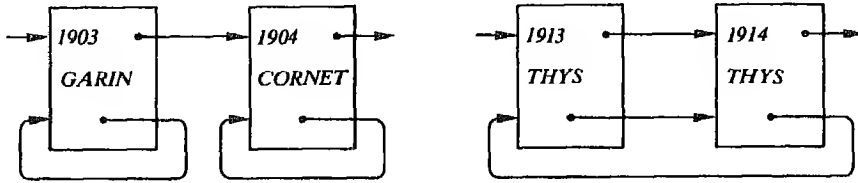
6.4 - أدلاء ومتغيرات تحريكية (Pointeurs et Variables dynamiques)

في حالات عديدة ، يمكن زيادة سرعة الحساب ، أو حتى إستبداله عن طريق تركيب ملائم للمعطيات . لنفرض مثلاً قائمة رابحي دوري فرنسا للدراجات والأسئلة :

- (أ) - من ربح في السنة x ؟
 - (ب) - هل ربح مرات أخرى ؟
 - (ج) - في أية سنة ربح فلان ؟ إلخ . .
- إن تمثيلاً للمعطيات في جدول

tour : array [1903 .. 1990] of packed array [1..n] of char (دوري)

يسمح بإعطاء جواب سهل على السؤال (أ) (tour [x]) ، شرط إعتداد اصطلاح معيّن بالنسبة للسنين التي لا رابح فيها ، لكنه يجبر على تصفّح كل الجدول للإجابة على (ب) و(ج) . إن تمثيلاً مناسباً أكثر للأسئلة يمكن أن يكون على الشكل :



(GARIN, CORNET, THYS هم أسماء الرابحين في السنين المذكورة)
ويمكن وصفه في لغة الباسكال :

```
tour = array [ 0..87 ] of record
  annee: 1903..1990;
  nom: packed array [ 1..n ] of char;
  autre: -1..87
end
```

(annee : سنة ؛ nom : إسم ؛ autre : مختلف)

0	1903	GARIN	-1
1	1904	CORNET	-1
2	1913	THYS	3
3	1914	THYS	2

لكن تركيب المعطيات هذا ، لا يسمح بسهولة بزيادة فقرة (THYS-1920) ،
ويجبُ بالأخص الفكرة الأولية .

يمكن وصف العلاقة بين الفقرات بواسطة أدلاء : يسمح الدليل (pointeur)
بتسمية أداة . تُفَرَّق :

- إسم الدليل ، إنه معرفٌ لمتغير ؛

- قيمة الدليل : P

- الأداة المدلّل عليها : P ↑

يكون المتغير الدليل مُرتبط بنوع واحد :

```
type lien=^vainqueur;
vainqueur=record annee:1903..1990;
nom:packed array[1..n] of char;
autre : lien
suite : lien
end;
var tour : lien;
```

(vainqueur : متتصر ؛ lien : رباط ؛ suite : تابع) .

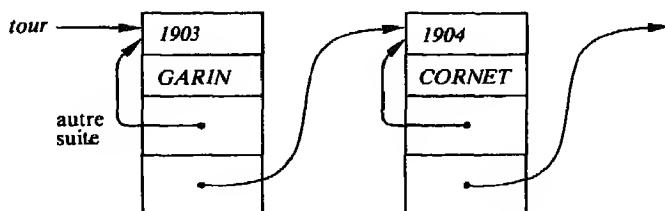
إنَّ Var tour : lien

تناسب مع التركيب المطلوب . يوجد علاقيتين بين الفقرات :

- autre (مختلف) ، يدلّ على إنتصار لنفس المتسابق ،

- unité (وُحدة) ، تدلّل على السنة التالية .

يدلّل المتغير tour (دوري) على رأس قائمة المتصرين .

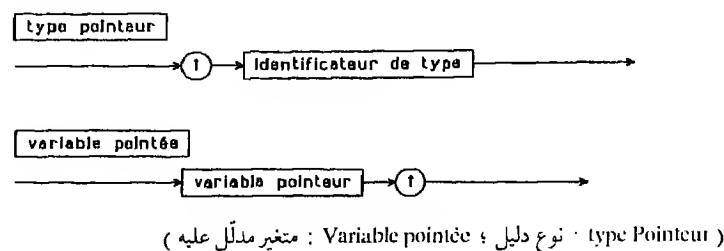


lien هو من النوع	tour
Vainqueur هو من النوع	tour↑
تساوي 1903	tour↑.année
له قيمة tour ↑	tour↑.autre↑
يساوي « CORNET »	tour↑.suite↑.nom

لا يتم التصريح عن الفقرات المدلل عليها كمتغيرات ، في القسم Var ، إنها متغيرات تحريكية ، الذي يجب خلقها صراحة (إجراء new) والذي يمكن إتلافها (إجراء dispose) .

new [p] يخلق متغيراً جديداً من نوع مدلل عليه بـ p ويعطي للدليل p قيمة تسمح بإسناد المتغير . يمكن إختبار مساواة دليلين مدللين على نفس النوع : العمليات = و > < ، إستعمل دليلاً في عملية تعيين (في القسم الأيسر أو الأيمن) وفي تعبير ، وحوله إلى وسيط . إن القيمة nil المُساوقة مع كل نوع دليل ، تشير بأن دليلاً لا يؤمن اسناداً لمتغير (إذا p : = nil ، يمكن بلوغ قيمة p (لكن ليس p ↑) ، بينما لا يمكن بلوغ قيمة p إذا كانت غير محددة قبل أية عملية تعيين) .

dispose (p) يُزيل المتغير المدلل عليه ؛ يعدّ خطأً إذا كان لـ p القيمة nil أو هو غير محدداً .

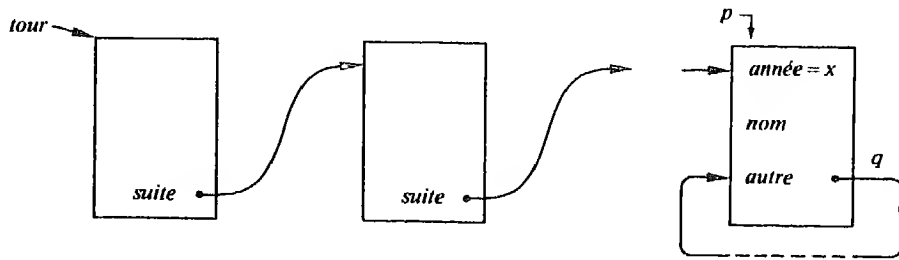


يُعرف النوع دليل قبل النوع المدلل عليه (إنه الشواذ الوحيد عن القاعدة » على كل معرّف أن يصرّح عنه قبل إستعماله ») .

ستتم الإجابة عن السؤال « من ربح في السنة x ، هل ربح مرات أخرى ؟ » بواسطة : (مع var p.q: lien ، وعلى أساس أن القائمة مرتّبة بالتنظيم التصاعدي للسنوات) .

```
p:=tour;
while (p^.annee<x) and (p^.suite<>nil) do p:=p^.suite;
if p^.annee=x then begin
  writeln(p^.nom, ' a gagne en ',p^.annee);
  q:=p^.autre;
  if q<>p do begin
    write('il a aussi gagne en ');
    while q<>p do begin
      write(q^.annee:5); q:=q^.autre end;
    writeln
  end
end
else writeln ('annee non referencee')
```

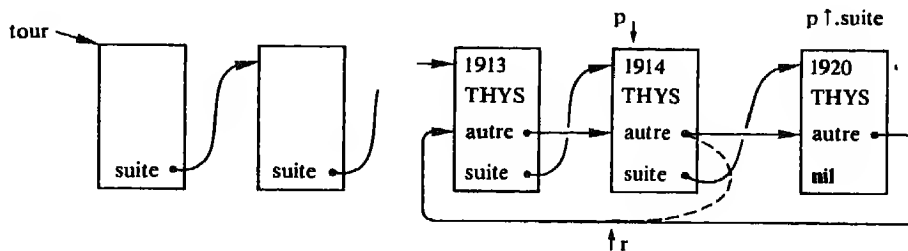
(a gagné en : ربح في ، aussi : كذلك ؛ année : سنة ؛ non referencee : غير مسندة)



بما أن القائمة تم تشكيلها ، فإن زيادة إسناد جديد لاحق على القائمة (سنة x ،
 اسم y) سيكون :
 (مع var p, q, r : lien) .

```
p:=tour; q:=p; r:=nil;
while q<> nil do begin { احث عن p = نهاية القائمة }
  if q^.nom=y then r:=q;
  p:=y;
  q:=q^.suite end;
new(p^.suite);
with p^.suite do begin { شغل قائمة الإنتصارات الأخرى }

  annee:=x; nom:=y; suite:=nil;
  if r=nil then
    autre:=p^.suite
  else begin
    autre:=r^.autre; r^.autre:=p^.suite end
end
```

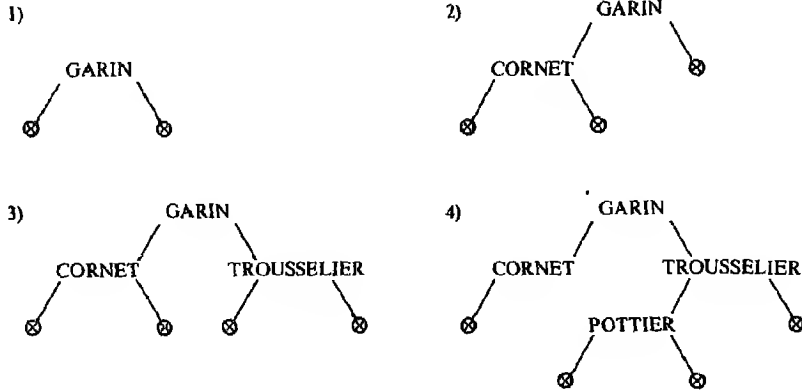


new (p, cl, ... cn) يخلق متغيراً جديداً من النوع فقرة مدلل عليها ، عن طريق
 إنتقاء الأقسام المشتقة المعلبة التي تتطابق مع الثوابت cl .. cn .
 dispose (p, cl... cn) يزيله ؛ يجب أن تكون المشتقات المشار إليها هي نفسها التي
 كانت عند الخلق .

مثال : فرز ثنائي (tri binaire)

تتلخص الطريقة بالحصول على تكوين شجري حيث تكون القيمة الموضوعية في
 عقدة أكبر من كل تلك التي تكون على اليسار وأصغر من كل تلك التي تكون على اليمين ؛

لنفرض أننا نريد فرز 'GARIN' ، 'CORNET' ، 'TROUSSELIER' ، 'FABER' ، 'PETIT-BRETON' ، 'POTTIER' ... إلخ .



تتمثل العلاقة بين العُقد بسهولة بواسطة أدلاء :

```
type fils=^noeud;
    noeud=record valeur:packed array [1..n] of char;
                gauche,droite:fils end;
var racine:fils;
```

إطبع القائمة المفروزة ، يعني
إطبع القائمة المفروزة لما هو على اليسار
إكتب القيمة المركزية
إطبع القائمة المفروزة لما هو على اليمين .

إنها عملية تكرارية والتي نوقفها عندما نعرف أن نطبع مباشرة ما هو باقي للطبع :
مثلاً القائمة الفارغة هي سهلة الطبع ؛ من هنا إجراء الطبع :

```
procedure editor(arbre:fils);
begin
    if arbre<>nil then begin
        editor(arbre^.gauche);
        writeln(arbre^.valeur);
        editor(arbre^.droite)
    end
end;
```

(éditer : إطبع ؛ arbre : شجر ؛ gauche : يسار ، droite : يمين ؛ valeur : قيمة) .

الذي سيتم تسميته في البرنامج بـ (éditer (racine).
 لزيادة قيمة u ، نجري مقارنات متتالية حتى الحصول على مكان حر

```
y:=racine;
repeat
  x:=y;
  if u<y^.valeur then y:=y^.gauche
  else y:=y^.droite
until y=nil;
if u<x^.valeur then creer(u,x^.gauche)
else creer(u,x^.droite)
```

مع الإجراء (خلق) créer

```
procedure creer(v:chaîne; var p:fils);
begin
  new(p);
  with p^ do begin
    valeur:=v; gauche:=nil; droite:=nil end
end;
```

```
program TriBinaire(input,output);
const n=20;
type chaîne=packed array [1..n] of char;
fils=^noeud; { شجراية ثنائية }
noeud=record valeur:chaîne;
gauche,droite:fils end;
var racine:fils;
c:chaîne;

procedure creer (v:chaîne;var p:fils);
begin new(p);
  with p^ do begin valeur:=v; gauche:=nil; droite:=nil
end
end; {creer}

procedure ajouter(u:chaîne);
var x,y:fils;
begin y:=racine;
  repeat x:=y;
    if u<y^.valeur then y:=y^.gauche
    else y:=y^.droite
  until y=nil;
  if u<x^.valeur then creer(u,x^.gauche)
  else creer(u,x^.droite)
end; {ajouter}
```

```

procedure editor(arbre:file);
begin
  if arbre<>nil then begin
    editor(arbre^.gauche);
    writeln(arbre^.valeur);
    editor(arbre^.droite)
  end
end; { اخلق }

begin { أضف }
  readln(c);
  creer(c,racine);
  while not eof do begin
    readln(c);
    ajouter(c)
  end;
  editor (racine)
end.

```

ملاحظة : تسمح مشتقات الفقرات والأدلاء في الباسكال (على أكثرية الحاسوبات) ، ببلوغ فكرة العنوان : إذا كان الدليل عنواناً ويشغل كلمة ، إذا شغل عدد صحيح كلمة ، عدد حقيقي إثنين وإذا شغلت كذلك مجموعة من 32 عنصراً كلمة ، فإذاً مع التصريح .

```

type ptr=^item;
genre=(bits,octets,mots,entiers,reels,addresses);
item=record
  case genre of
    bits: (bit:packed set of 0..31);
    octets: (octet:packed array[0..3] of 0..255);
    mots, entiers: (i:integer);
    reels: (r:real);
    addresses: (a:ptr)
  end;
var mem:item;

```

(bits : بتات ؛ octet : بايتات ؛ mots : كلمات ؛ entiers : أعداد صحيحة ؛ réels : أعداد حقيقية ؛ addresses : عناوين) .

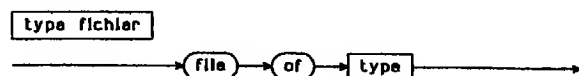
نبلغ في الذاكرة مثلاً البتة x من الكلمة y :

$mem.i := y$	بلوغ
$x \text{ in } mem.a \uparrow .bit$	إختبار
$mem.a \uparrow .bit := mem.a \uparrow .bit + [x]$	ضبط بـ 1
$mem.a \uparrow .bit := mem.a \uparrow .bit - [x]$	ضبط بـ 0

7.4 - سجلات (Fichiers)

تتطابق فكرة السجل مع مسلسل مركبات ، كلها من نفس النوع ، ذوات عدد غير محدد مسبقاً (هذا ما يفرقه عن الجدول) . يمكن بلوغ مركب واحد في وقت واحد ؛ نبُلّغه بواسطة نافذة نزيحها عن طريق إستعمال الإجراءات المعرفة مسبقاً `get` في الشأن معانية ، و `Put` في الشأن نتائج . يمكن كذلك إعادة وضع النافذة على بداية المسلسل بواسطة `reset` للمعانية التسلسلية ، أو إتلاف المسلسل لنخلق منه مسلسل جديد بواسطة `rewrite` للنتائج التسلسلي . أخيراً عندما تصل النافذة الى نهاية المسلسل ، فإن الشرط `eof` يصبح صحيحاً .

عند تصريح السجل ، نحدد نوع المركبات



مثال : `var f : file of integer`

يمكن بلوغ المركب الراهن في السجل بواسطة النافذة أو المتغير الداريء ، المرافق للسجل :



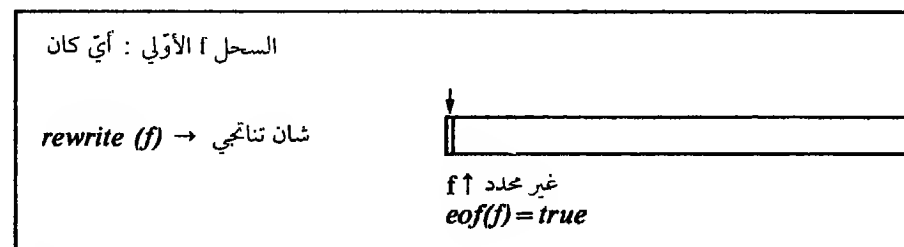
(variable tampon : متغير داريء ؛ fichier : سجل)

مثال : `f ↑ = 12` (لِعَدَم خَلِطَه مع المتغير المدلل عليه)

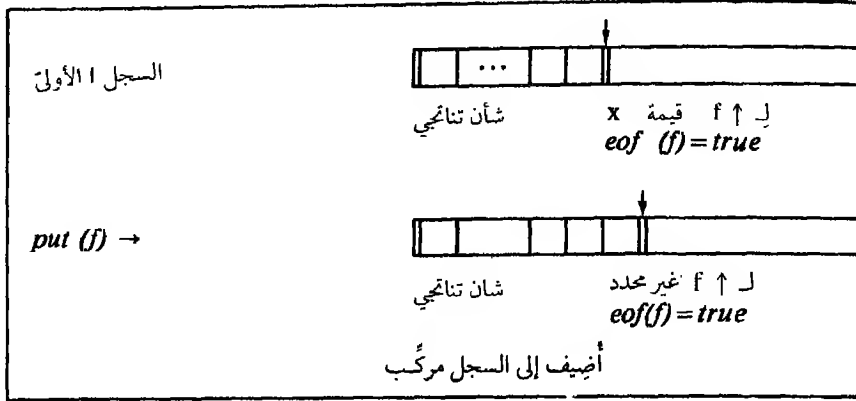
`rewrite (f)` يضع السجل في الشأن نتائج

`eof (f)` يساوي `true` ، السجل فارغ (ضاعت قيمته القديمة) و `f ↑` غير

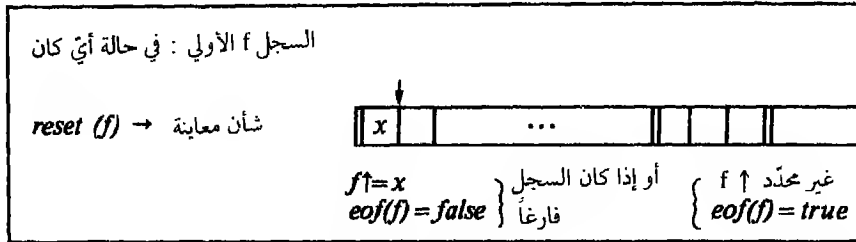
محدد : يجب إعطائه قيمة قبل كل عملية `put (f)`



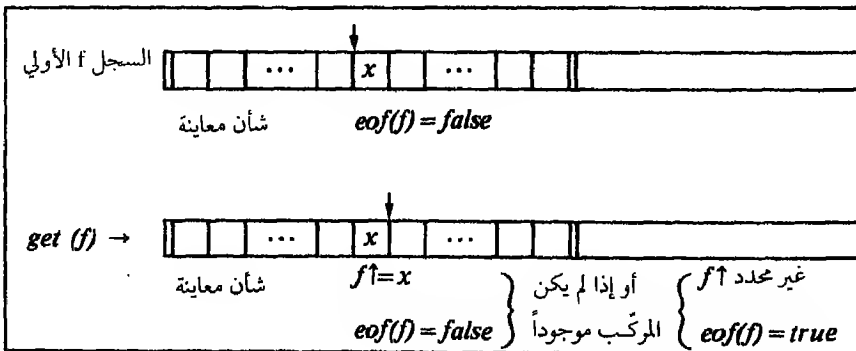
$put(f)$ يضيف قيمة $f \uparrow$ في نهاية السجل ، الذي يجب أن يكون شأناً تناهجياً ، ويقدم النافذة .



$reset(f)$ يضع السجل في الشأن معانية
إذا لم يكن السجل فارغاً ، $eof(f)$ تساوي false و $f \uparrow$ تساوي المركب الأول (تنفذ reset أول ger ضمني)



$get(f)$ يقدم النافذة وإذا وُجد المركب ، عيّنه به $f \uparrow$ ؛ يجب أن يكون السجل في الشأن معانية .



مثال : نسخ copie

إنسخ السجل f على السجل g ، السجلين من نفس النوع

```

program copie (f,g);
{ نسخ لسجلات خارجية عن طريق استعمال put و get }
type sequence=file of record nom:integer; qte:real end;
var f,g:sequence;
procedure copie (var u,v:sequence);
begin
  reset(u); {reset remplit le fenetre u^}
  rewrite(v);
  while not eof(u) do begin
    v^:=u^; put(v); get(u) end
  end;
begin copie(f,g) end.

```

المعرفان f و g هما خارجيان عن البرنامج (أنظر 2.2) ؛ لا يمكن أن يكون السجل وسيط قيمة .

مثال تمّدد Etendre

يُحضّر هذا الإجراء ، السجل f من النوع T ، لعملية إلحاق مركّبات ، لن نضع أي افتراض على الحالة الأولى لـ f .

```

procedure PrepareEtendre(var f:T);
var auxiliaire:T;
procedure copie (var u,v:T);
begin
  reset(u); {reset remplit le fenetre u^}
  rewrite(v);
  while not eof(u) do begin
    v^:=u^; put(v); get(u) end
  end;
begin copie(f,auxiliaire); copie(auxiliaire,f) end;

```

إن السجل auxiliaire (مساعد) هو موضعي في الإجراء : يُخلَق عند تنشيطه ويُتلف عند الرجوع إلى المنادى .

إختصارات

فيما عدا السجلات text :

1 - read (f, x) مُكافئ لـ

begin x := f ↑ ; get (f) **end**

x هو متغير يمكن تعليقه

2 - read (f, x1, ..., xn) مُكافئ لـ

begin read (f, x1) ; ... read (f, xn) end

3 - write (f, x) مُكافئ لـ

begin f ≤ 7 : = x ; Put (f) end

x هو تعبير

4 - write (f, x1, ..., xn) مُكافئ لـ

begin write (f, x1) ; ... ; write (f, xn) end

مثال : نسخ

```
procedure copie2(var u,v:sequence);
var x:....; {composant de sequence}
begin
  reset(u); rewrite(v);
  while not eof(u) do begin
    read(u,x);      {x:=u^; get(u)}
    write(v,x)      {v^:=x; put(v)}
  end
end;
```

يُعني الشكل المختصر read / write من المعالجة المملة للنوافذ ، لكنه يخفي عن قارئ البرنامج العمل الدقيق :
يُستعمل get بعد إستعمال النافذة
يُستعمل eof قبل إستعمال النافذة .

تناسب إذن العبارة طالما (tant que) وبشكل جيد مع توقيف مع إقتصار :

إنتاج العنصر الأول ← reset (u)
while شرط على العنصر المنتج do
عَالِج العنصر ← u ↑
أنتج العنصر التالي ← get (u)

بعد تنفيذ read (u, x) ، تحتوي النافذة u ↑ على المركب التالي المُحصّل عليه بـ read ؛ هذا ما يشكّل فائدة كبيرة في كثير من المسائل ذات الطابع التحليلي النحوي ، التي تتطلب معرفة مسبقة للمركب .

مثال : فرز - إندماج

يتمّ الفرز في ذاكرة أساسية ، أي في جدول ، بينما تكون المعطيات في ذاكرة ثانوية ، ممثلة غالباً بسجل خارجي . غير أن الذاكرة الأساسية لها حجم محدد ، هذا ما يحدّ من عدد القيم التي يمكن فرزها (يتوقف هذا العدد على الحاسوب المستعمل ، يمكن أن يكون

1000 أو 10 000 على ميكروحاسوب ، أو يتعدى المليون على حاسوب كبير) .

تتلخص الفكرة الأساس في الفرز - إندماج ، بتقطيع المعطيات إلى N سجل من P عنصر ، بشكل يمكن من فرز P قيمة في الذاكرة ، N مرة متتالية ، ومن ثم دمج السجلات المفروزة .

لنفرض أننا نريد خلق سجل h ناتج عن إندماج السجلين f و g المفروزين بالترتيب التصاعدي :

```
type fichier=file of integer;

procedure fusion(var f,g,h:fichier);
{ دمج السجلات المرتبة f و g بسجل واحد h ، هو أيضاً مرتب }

var vide:boolean;
begin
  reset(f); reset(g); rewrite(h);
  { 1 دمج حتى نهاية السجل }
  vide:=eof(f) or eof(g);
  while not vide do begin
    if f^<g^ then begin
      h^:=f^; get(f); vide:=eof(f) end
    else begin
      h^:=g^; get(g); vide:=eof(g) end;
    put (h)
  end; {while}
  { 2 / نسخ للنهية المحتملة للسجل f }
  while not eof(f) do begin
    h^:=f^; get(f); put(h) end;
  { 3 / نسخ للنهية المحتملة للسجل g }
  while not eof(g) do begin
    h^:=g^; get(g); put (h) end
end; { إندماج }
```

8.4 - سجلات النص (Fichiers de Texte)

إن سجل النص هو سجل سمات مركّب على هيئة سطور ؛ تُطبّق الإجراءات والدوال reset ، rewrite ، put ، get ، eof ، read ، write على معالجة السمات في سجل نص بنفس الطريقة التي تطبق فيها على سجل سمات .

يُعبّر المعرّف المعرّف سابقاً text عن نوع سجل النص . بفعل تركيبه على هيئة سطور ، فإن أربع عمليات إضافية هي متاحة :

1 - writeln (f) يضع علامة نهاية السطر في السجل f ؛ لا تكون علامة نهاية السطر ، عامة ، سمة خاصة : إنها تتوقف على الحاسوب المُستعمل ، لكن يتم

إعادة قراءتها (بـ get أو read) كسمة تباعد .

writeln (f, e1, .. en) مُكافئ لـ

begin write (f, e1, en) ; writeln (t) end

2 - readln (f) له الأثر بوضع الموقع الجاري للسجل f مباشرة بعد نهاية السطر الجاري فيه العمل ؛ تجد النافذة ↑ f هكذا نفسها مركزة على السمة الأولى للسطر التالي في حال وجوده .

readln مُكافئ لـ

begin read (f, v1, ..., vn) ; readln (f) end

3 - تُرجع الدالة البولية eoln (f) القيمة true إذا كانت النافذة ↑ f مركزة على نهاية سطر ؛ في هذه الحالة = ↑ f ، ووحدها eoln (f) تسمح بالفريق ما بين تباعد نهاية السطر هذا والتباعد الإعتيادي .

4 - page (f) يؤدي إلى طباعة النص الذي يلي على صفحة جديدة ، عندما يكون السجل f مطبوعاً على جهاز ضوئي ملائم .

إختصارات : إذا أغفل ذكر إسم السجل ، يُطبّق الإجراء أو الدالة على إحدى السجلات المعرفة مسبقاً input أو output :

<i>write (output, e)</i>	تعني	<i>write (e)</i>
<i>writeln (output)</i>		<i>writeln</i>
<i>read (input, v)</i>		<i>read (v)</i>
<i>readln (input)</i>		<i>readln</i>
<i>eoln (input)</i>		<i>eoln</i>
<i>eof (input)</i>		<i>eof</i>
<i>page (output)</i>		<i>page</i>

read و write يقبلان كذلك بوسائط أخرى غير النوع سمة ؛ تمّ تفصيل المعالجة في الفقرة « دُخِل - خُرج » في 4.2 .

مثال : نريد إعادة كتابة نص ، مؤلف من كلمات مفصولة بتباعدات ، على عرض معطي ، دون تقطيع للكلمات (إلا إذا كانت الكلمة كبيرة جداً لتكتب على سطر واحد) .

1 - السجل i الحاوي للنص الأولي سيتم تصفحه كلمة بعد كلمة :

```

while not eof (i) do begin
  sauter les espaces superflus précédant un mot
  lire un mot      c
  s'il ne tient pas sur la ligne en cours
    aller à la ligne
  écrire le mot
end

```

```

Sauter les espaces superflus précédant un mot
lire un mot
S'il ne tient pas sur la ligne en cours
  aller à la ligne
écrire le mot
end

```

تخطى التباعدات الزائدة السابقة لكلمة
 اقرأ كلمة
 إذا لم يكن السطر الجاري العمل فيه
 إلى سطر جديد
 اكتب الكلمة

2 - « تخطى التباعدات الزائدة » :

```
repeat read(i,c) until c<>' '
```

Var c : char مع

3 - « اقرأ كلمة » : نقوم بترتيبها في جدول كلمة (mot)

```
array [ 1... long ] of char
```

حيث يكون الثابت long ، الطول الأقصى لكلمة تُكتب على سطر ، أي حجم السطر ؛ يعطي الدليل m موقع السمة الأخيرة المدخلة في الجدول ، تساوي m إذن صفراً في البدء . تتم القراءة سمة بعد سمة ؛ الأخيرة المقروءة ، تباعداً ، لا تدخل ضمن الكلمة : تكرارية مع إقتصار ← while

```

b:=0;
while c<>' ' do begin
  m:=m+1; mot[m]:=c; read(i,c)
end

```

→ إستعمل
 → انتج

تم إنتاج الحد الأول c ب « تخطى التباعدات »

4 - « إذا لم يكن السطر الجاري العمل فيه ، إلى سطر جديد » ، يستعين بالموقع الأول الحر على السطر ، 1 (في البدء) :

```
if (1+m-1)>long then begin writeln(f); 1:=1 end
```

5 - « اكتب الكلمة »

كان السطر كافياً للكلمة ، نكتبها ، ثم نسعى لتحضير كتابة الكلمة التالية عن طريق فصلها عن الكلمة المكتوبة بواسطة تباعد .

```

for x:=1 to m do write(f,mot [x]);
l:=l+m;
if (l+l)>long then begin writeln(f); l:=1 end
else begin write(f,' '); l:=l+1 end

```

٦ - لكن إذا كان طول الكلمة أكبر من طول السطر ، يجب تقطيعها ، من هنا التدقيق في « إقرأ كلمة »

```

m:=0;
while c<>' ' do begin
  m:=m+1; mot[m]:=c; read(i,c);
  if (m=long) and (c<>' ') then begin
    {mot>ligne,le couper}
    if l<>1 then writeln(f);
    for x:=1 to long-1 do write(f,mot[x]);
    writeln(f,' '); l:=1; mot[l]:=mot[m]; m:=1
  end
end
end

program reecrire(i,f);
{ أعد الكتابة على النص مقروء على المؤلف من }
{ كلمات مفصلة بتعدادات بشكل لا تتعدى طول معطي لكل سطر }
const long=36; { طول السطر في النص النهائي }
var i,f:text; { النص الأولي ، النص النهائي ، سجلات خارجية }
    l:integer; { الموقع التالي الحرفي السطر }

    mot:array [1..long] of char; { الكلمة المقروءة }
    m:integer; { دليل في كلمة آخر سمة مُدخلة }

    c:char; { سمة في الدخل }
    x:integer;
begin
  reset(i); rewrite(f);
  l:=1;
  { تصفح النص الأولي كلمة كلمة }
  while not eof(i) do begin
    m:=0;
    { تخطي التباعدات الزائدة السابقة للكلمة }
    repeat read(i,c) until c<>' ';
    { أنه قراءة الكلمة }
    while c<>' ' do begin
      m:=m+1; mot[m]:=c; read(i,c);
      { إذا كانت الكلمة طويلة بالنسبة للسطر ، قطعها }
      if (m=long) and (c<>' ') then begin
        if l<>1 then writeln(f);
        for x:=1 to long-1 do write(f,mot[x]);
        writeln (f,' '); l:=1; mot[l]:=mot[m]; m:=1
      end
    end
  end
end

```

```

end;
{ إذا كانت الكلمة المقروءة طويلة بالنسبة لنهاية السطر الجاري العمل فيه ، إنتقل الى سطر جديد }

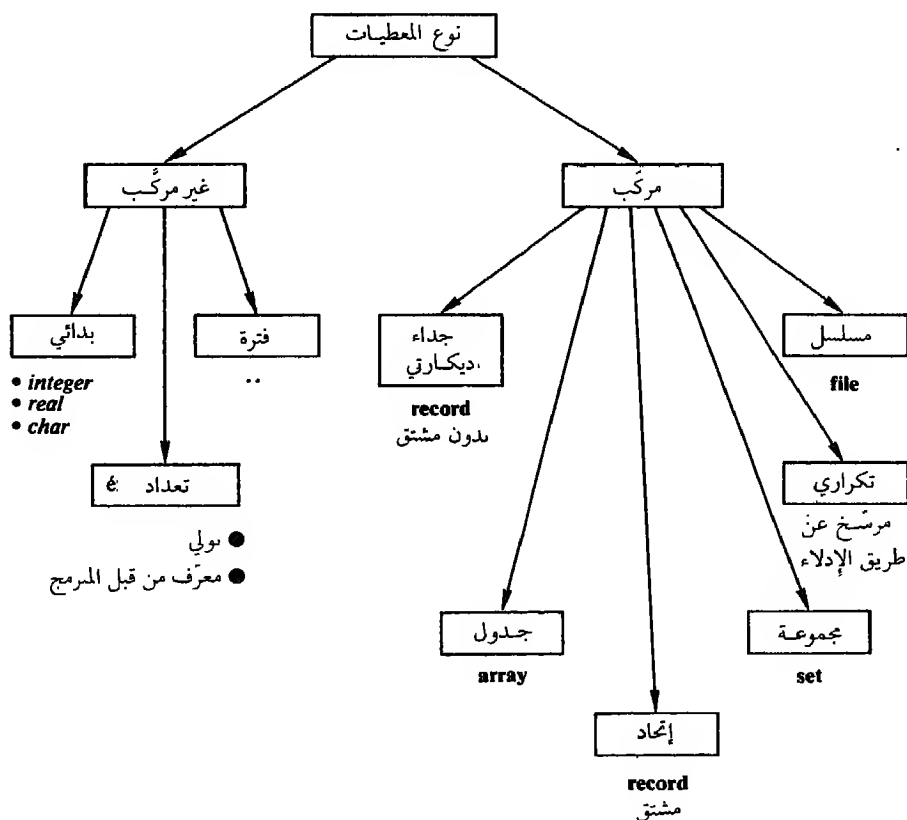
if (1+m-1)>long then begin writeln(f); 1:=1 end;
{ إكتب الكلمة }
for x:=1 to m do write(f,mot[x]);
1:=1+m;
{ حفّض لكتابة الكلمة التالية : ضع تباعداً لفصلها عن الكلمة السابقة }

if (1+1)>long then begin writeln(f); 1:=1 end
else begin write(f,' '); 1:=1+1 end
end; {while not eof (1)}
if 1>1 then writeln(f)
end.

```

9.4 - التسلسل العشيري للأنواع (hiérarchie des types)

ترجع لغة الباسكال ، المعرفة من قبل ن. ويرث (N. Wirth) إلى فكرة البرمجة المركبة ؛ على الأخص أنواع المعطيات الموافقة لترسيخ معنى (مقترحة من ث. أ. ر. هوار (C.A. R. Hoare) .



10.4 - متممات (compléments)

أعداد شبه صدفية (Nombres pseudo-aléatoires)

لا تقدّم لغة الباسكال صراحة مولداً لأرقام شبه صدفية . فيما يلي بعض التقنيات ،
اللازمة لإنجاز مولد :

توزيع متمائل على $[1, 0]$

```
function slea(var germe:integer):real;
begin
  alex:=germe/65535;
  germe:=(25173*germe+13849) mod 65536
end;
```

تكوّن هذه الدالة 65 536 قيمة صدفية مختلفة ، موزعة بشكل متمائل على $[1, 0]$
قبل أن تتكرّر . إنها تعمل على كل حاسوب يكون فيه $1 - 2^{31} \leq \text{maxint}$. يجب أن يُحفظ
الوسيط المتغير germe (بذرة) من نداء إلى نداء .

لـ $1 - 2^{15} \leq \text{maxint}$ ، يمكن إستعمال

```
function unif(inf,sup:real; var germe:integer):real;
begin
  germe:=germe*899;
  if germe<0 then germe:=germe+32767+1;
  unif:=germe/32767.0*(sup-inf)+inf
end;
```

التي تولّد قيمتها على $[$ أدنى ، أقصى $]$.

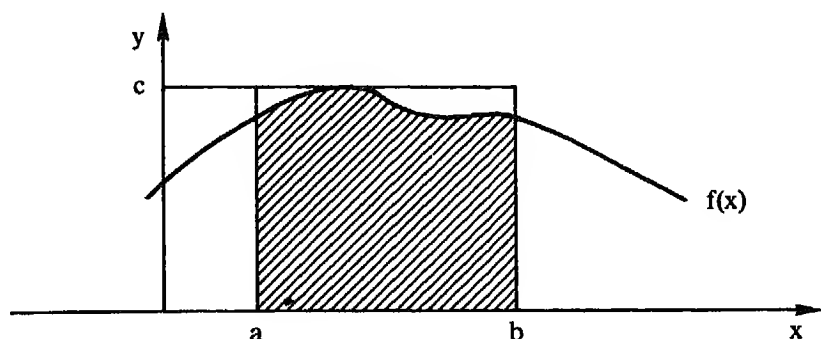
توزيع غوسي (Répartition gaussienne)

```
function Gauss(moyenne,ecartType:real;
               var germe:integer):real;
var k:integer; S:real;
begin
  S:=0;
  for k:=1 to 12 do S:=S+unif(0.0,1.0,germe);
  Gauss:=moyenne+(S-6.0)/12.0*ecartType
end;
```

(moyenne : متوسط ، écart type : إنحراف معياري)

مثال : مُكاملة على طريقة مونت - كارلو - Intégration par la méthode de Monte-Carlo.

لنفرض أننا نريد حساب مكاملة $y = f(x)$ على الفترة $[a, b]$ حيث $f(x)$ حد أقصى c :



لحساب المساحة، $\int_a^b f(x) dx$ سنعمل على الاختيار بالقرعة لعدد كبير N من النقاط x, y في المستطيل $a \leq x \leq b$ و $0 \leq y \leq c$ ؛ إن نسبة العدد P للنقاط الواقعة ضمن المساحة $(y \leq f(x))$ إلى العدد الإجمالي للنقاط يقارب نسبة التكامل الى مساحة المستطيل :

$$\frac{P}{N} \xrightarrow{N \rightarrow \infty} \frac{\int_a^b f(x) dx}{(b-a)c}$$

```

program MonteCarlo(output);
var germe:integer;
    N,P,i:integer;  a,b,c,x,y:real;
    function rand:real;
    begin
        germe:=germe*899;
        if germe<0 then germe:=germe+32767+1;
        rand:=germe/32767.0
    end;
begin
    N:=10000; germe:=17; P:=0; a:=0.0; b:=4.0; c:=16.0;
    for i:=1 to N do begin
        x:=a+rand*(b-a);  y:=rand*c;
        if y<=sqr(x) then P:=P+1
    end;
    writeln('integrale de x*x sur [0,4]');
    writeln('valeur analytique: ',b*b*b/3.0:8:5);
    writeln('valeur approchee: ',c*(b-a)*P/N:8:5)
end.

```

(valeur : قيمة ؛ analytique : تحليلي ؛ approchée : تقريبية)

التثلم Indentation

التثلم ، الذي هو فعل تصريح سطر ، ضروري بالنسبة لمقروئية البرامج :

```

if x < T[c] then
b:=c
else
a:=c;

```

```

if x < T[c] then
b:=c
else
a:=c;

```

نحاول عن طريق التثلم ، تجميع سطور البرنامج الموجودة في نفس المستوى المنطقي . هذه هي حال التعريفات ، التصريحات Procedures Var, type, const, label و function والعبارات المركبة .

```

begin
E1;
E2;
...
En
end

```

```

if c then E1
else E2

```

```

while c do E

```

```

if c then begin
E1;
E2;
...
En

```

```

while c do begin
E1;
...
En
end;

```

```

end else begin
Ep;
...
Em
end

```

```

for i:=d to a do E

```

```

repeat
E1;
...
En
until c

```

```

for i:=d to a do begin
E1;
...
En
end

```

etc.

يتوجّه التثلم للقارئ الإنساني وليس للآلة : لا يوجد غط واحد لعرض البرامج .

البرنامج المقروء هو المثلم جيداً والذي يحتوي على أجزاء كثيفة بقدر كاف مفصولة بسطور بيضاء ؛ لا يجب التردد في وضع الملاحظات .

سهولة النقل (Portabilité)

إن البرنامج الذي يكون عمله مرضٍ بشكل كامل ، غالباً ما يتم نقله على حاسوبات أخرى غير الحاسب الذي كتب فيه . يمكن أن يكون النقل مباشراً أو شديد الصعوبة تبعاً للطريقة التي بها كُتب البرنامج . تهدف التوصيات الذاتية فقط إلى تحسين سهولة النقل هذه للبرنامج وليس لتعريف ما يكون « البرنامج الجيد » (« البرنامج الجيد » يرضي تماماً مستعمليه) .

* لا تستعمل إلا السمات المعروفة من اللغة الموحدة :

أرقام ، أحرف ، تباعد ، سمات خاصة :

> () ↑ : ؛ . [] = / * - +
{ } . <

لكن - لا تختَر طريقة شاذة لكتابة الأحرف (أحرف كبيرة ، أحرف كبيرة وأحرف صغيرة)
- لبعض السمات تمثيل متناوب :

{ } ↑ []
(★) (★) ^ (. .)
@

يجب عدم إستعماله إلا في حال عدم صلاحية سمات الإسناد .

* لا تفترض مزايا خاصة للعب السمات :

'a' < 'b' < ... < 'z'

'0' < '1' < ... < '9'

ord ('n') - ord ('0') = n 0 ≤ n ≤ 9

مثلاً succ ('c') أو '>' . . '+' ليسوا كتابات سهلة النقل .

* لا تستعمل إلا الكلمات الدلالية والمعرفين المعرفين مسبقاً في النظم (أنظر الملحق 3) ؛
بالأخص لا تستعمل مشتقات وطنية .

* لا تستعمل تمّدداً للغة حتى ولو بدا ضرورياً : لكل حالة في الباسكال تمّدداتها ، غير المتساوقة مع الحالات الأخرى .

* لا تقم بإفتراضات متفائلة حول دقة الأعداد الحقيقية ، حقول الطبع الغيائية ، عدد السمات ذات المدلول لمعرف (إنها في بعض الأحيان 8 ، رغم النظم) ، حجم مجموعة (الإقتصار على 59...0 هو معقولاً ! ؛ النوع Set of char ليس دائماً صالحاً) .

* صرّح في بداية البرنامج عن الثوابت لكل القيم العددية أو الأبجعددية المستعملة .

* إعزل وفسّر أجزاء البرنامج المتعلقة بالحاسوب (نيل العناوين ، سجلات مباشرة

إسنادات خارجية ، خيارات التصريف ، اعداد ثمانية وسادس عشرية ، إلخ
 . (...)

11.4 - تمارين

- 1 - اكتب كل التبديلات (Permutations) لكلمة من n حرف (يوجد ! n (عاملي n) ؛
 مثلاً 3 = N DES DSE EDS ESD SED SDE) .
- 2 - لكي نكوِّد نصاً ، سنستبدل كل حرف باللاحق في الألفباء ('A' هو لاحق 'Z') ؛
 اكتب إجراءات التكويد ونزع الكود .
- 3 - حوِّل عدد صحيح (< 1000) بالأحرف الكاملة .
- 4 - يساوي العدد الصحيح الكامل مجموع قاسميه ، يدخل في ذلك 1 بينما لا يدخل العدد
 نفسه . إحصب الأعداد الكاملة الأصغر من 500 .
- 5 - اكتب الـ n أوَّل سطر من مثلث باسكال :

$$\begin{array}{cccc} & & 1 & 1 \\ & & 1 & 2 & 1 \\ & 1 & 3 & 3 & 1 \\ & & & & \dots \end{array}$$

(C_i) عندنا i ، العמוד j ،

- 6 - لنفترض معطياً تاريخ من الماضي على شكل نهار - شهر - سنة ، احسب نهار الأسبوع
 الموافق . نذكر بأن التقويم الغريغوري حلَّ مكان التقويم القيصري في العام 1582 ،
 وبأن سنة ألفية تُقسم على 4 هي كبيسة ، ما عدا السنوات الألفية 00 حيث يكون
 رقم القرن لا يقسم على 4 .
- 7 - نريد طبع جدول كلمات موجودة في نص على أن يطبع بالنسبة لكل كلمة ، قائمة أرقام
 السطر الذي يحويها .
- 8 - إقرأ تعبيراً واحسب قيمته ، في لغة يكون المعرف فيها حرفاً . لا يوجد سوى النوع
 حقيقي والعمليات + - * / ، كذلك المزدوجات ، مع قواعد الأسبقية المعتادة .
 مسبقاً سنعيّن قيمة لكل متغيّر .
- 9 - إطبّع جدولاً للـ 100 أوَّل عدد عشري من e ، قاعدة حساب اللوغاريتمات

$$\left(e = \sum_{n=0}^{\infty} \frac{1}{n!} \right)$$

- 10 - تلوين خارطة . لنفترض معرفة الحدود المشتركة بين عدة بلدان ، احسب اللون الذي
 يجب إعطاؤه لكل منهم (أربعة ألوان تكفي) بشكل أن بلدين لهما حدود مشتركة لا
 يكنى لهما نفس اللون .

11 - كَوْن متسلسلة من 100 عدد ، ثمّ في $[0, 1, 2]$ بشكل أن متسلسلتين ثانويتين متجاورتين لا تكونا متشابهتين .

مثال : ... 0102102 و ... 01021010 لا يلائمان .

12 - ضع تماني ملكات على رقعة داما (4×4) بشكل أن ملكتين لا تكونا على احتكاك مباشر (تبعا لقواعد لعبة الداما) .

الفصل الخامس

مذكّرة مساعدة

0.5 - برنامج، فِدرَة، مَدَى، منطقة Programme, Bloc, Portée, Région

0) - يتألف البرنامج من :

عنوان Program ...

تصريحات وتعريفات : (أنظر 2)

حسم begin ... end

منطقة

فِدرَة

1 - تتبع التصريحات والتعريفات النسق الإلزامي :

Label تصريحات الوسومات

Const تعريف الثوابت

type تعريفات الأنواع

var تصريحات المتغيرات

procedure
function } تصريحات الإجراءات والدوال

كل قسم هو اختياري

2 - يُكتب العنوان :

program (input, output) إسم

لبرنامج يقوم بأعمال القراءة والكتابة .

3 - يتألف تصريح الإجراء (أو الدالة) من :

عنوان .. procedure

تصريحات وتعريفات : (أنظر 2)

حسم begin .. end

;

فِدرَة

- 4 - تكون منطقة المعرف ، القدرة التي تم التصريح عنه فيها ، كذلك الفدرات التي تحتويها هذه القدرة .
- 5 - يكون مدى المعرف ، منطقته مطروح منها مناطق المعرفين الذين لديهم نفس كتابة الكلمات المصرح عنهم في فدرات داخلية .
- 6 - لا يمكن إستعمال المعرف إلا ضمن نطاق مداه .
- 7 - لدى المعرفين المعرف عنهم مسبقاً ، منطقة تحوي البرنامج ؛ يمكن إعادة التصريح عنهم .

8 - يجب التصريح عن كل معرف قبل إستعماله .

1.5 - معرف ، رمز ، فاصل (Identificateur, Symbole, Séparateur)

(0) يتألف المعرف فقط من أحرف وأرقام ، ويبدأ بحرف : B52 pi2 vazy

- 1 - سماته كلها لها مدلول
- 2 - سيان إستعمال سمات كبيرة ، صغيرة ، أم غليظة ، إلخ . . . : aBc Abc abc هم نفس المعرف .
- 3 - لبعض الرموز تمثيلات متناوبة :

[] { } ^ ou @
(.) (★ ★)

- 4 - بين كلمتين دليليتين معرفين ، ثوابت ، يجب على الأقل وجود فاصل واحد : goto 13 هي غير سليمة .
- 5 - لا يمكن وجود فاصل داخل معرف ، كلمة دليلية أو رمز : go to هي غير سليمة .
- 6 - الفواصل هي التباعد ، نهاية السطر ، الملاحظة .
- 7 - تكتب الملاحظة { ملاحظة } (Commentaire)
- 8 - لا يمكن إستعمال الكلمات الدليلية (... with, begin, var) كمعرفين .

2.5 - متغيرات (variables)

- (0) - إن المتغير هو كناية عن موقع في ذاكرة الحاسوب مخصص لإحتواء قيمة .
- 1 - تكون القيمة الأولية غير محددة .
- 2 - يسمح معرف لمتغير ببلوغ متغير .
- 3 - يجب أن تكون القيم المعينة لمتغير ، من نوع محدد ، ملاصق للمتغير .
- 4 - نحى تصريح المتغير ، متغيراً يربط به نوعاً ومعرفاً : var i : integer; x, y : char;
- 5 - يجب التصريح عن كل معرف لمتغير مُستعمل في فدر : داخل هذه القدرة (متغير موضعي) أو في فدر شاملة (متغير إجمالي) .
- 6 - يجب أن يتم إختيار المعرف بشكل يعكس دور المتغير الملعوب في البرنامج :

7 - يجب أن يتمّ التصريح عنه بملاحظة تحدد صراحة دوره .
 8 -

مثال	يتم وسمه إذن بـ	يمكن أن يكون المتغير
i	(identificateur)	معرف كامل
T [i , j]	(identificateur) [indice]	معرف [دليل] مكوّن لجداول
X. réel	(identificateur, champ)	معرف . حقل حقل فقرة
(صلة) lien ↑	(pointeur ↑)	دليل ↑
input ↑	(fichier ↑)	سجل ↑

وكل تأليف ، تبعاً an . [mort] . date ↑ . Père [NoSS] . individu

للنوع : (شخص [NoSS] . أب ↑ . تاريخ [موت] . سنة)

3.5 - أنواع (types)

0 - يصف النوع القيم الممكنة لأداة وطريقة نيل .

1 - يربط معرف النوع ، معرفاً بنوع

type complexe = record réel, imaginaire: real end;

$N = 0 \dots \maxint;$

(complexe : عُددي ، réel : حقيقي ، imaginaire : تحلي)

2 - إن تعريفات النوع هي ضرورية للتصريح عن وسائط إجراء أو دالة ، نتيجة دالة ، دليل (pointeur) ، ولتأمين تساوق المتغيرات :

type T = .. Var X : T ...

procedure (A : T) ...

A (X)

3 - لكل قيمة من نوع ترتيبي يتوافق عدد ترتيبي صحيح ؛ يتوافق ترتيب القيم مع ترتيب الأعداد الترتيبية .

4 - الأنواع الترتيبية هي : صحيح ، سمة ، بولي ، تعداد وفترة من نوع ترتيبي .

5 - بالرغم من كون النوع الحقيقي غير مركب ، فإنه ليس نوعاً ترتيبياً .

6 - نطبق على النوع الترتيبي عمليات العلاقة $< =$ ، $<$ ، $> =$ ، $>$ ، الدوال succ و pred (إنتقال إلى السلف أو الخلف) ، الدالة ord (التي تعطي العدد الترتيبي) . فيما عدا السمات ، فإنه ليس لـ ord دالة معاكسة .

7 - يكون النوع الترتيبي ضرورياً لتكوين فترة ، جدول (دليل) أو مجموعة .

8 - تبنى الأنواع المركبة (جدول ، سلسال ، فقرة ، مجموعة ، سجل) على الأنواع

البسيطة (حقيقي ، ترتيبى) والنوع دليل (type pointeur) .

4.5- ثوابت ، اعداد حقيقية ، اعداد صحيحة ، سلاسل ، Entiers, réels, constantes) (Chaines)

(١) - الثابت هو قيمة غير قابلة للتعديل من قبل البرنامج .

١- يَربُطُ تعريف الثابت ، معرفاً بقيمة :

```
const pi = 3.14159; zéro = '0'; n = 17; m n = - n;
```

2- يُكْتَبُ الثَّابِتَ الْحَقِيقِيّ مِنَ النُّوعِ real ، عَلَى شَكْلِ عِشْرِيّ (تَحْلُ النُّقْطَةِ مَكَانَ

3.2E1 : : 10^x يعني Ex) أو على شكل أسّي (الفاصلة) : 0.198 - 0.3
 0.43E-2 - 61E0

3- يكتب الثابت الصحيح من النوع integer ، بدون جزء عشري ولا أس : 32 61 - .
يجب أن تبقى قيمته ضمن الفترة maxint .. maxint - ، حيث يكون maxint ثابت
معرف مسبقاً في كل حاسوب .

4- يُكتب الثابت السلسلة ذي الـ n سمة من النوع `Packed array [1..n] of char` ، بين علامتي حذف : 'chaine' ('سلسلة') .

لكل السمات بما فيها التباعد ، كذلك تمثيلهن الحرفي ، معنى .
 الثابت السمة هو سلسل من سمة واحدة '+'
 في الثابت السلسل ذي ال n سمة ، تكرر علامة الحذف :
 ' le fond de l'air est frais '

5- يُعدّ الثابت nil متساوياً مع كل الأنواع أدلاء (pointeurs)

7- يمكن استعمال معرف الثابت في كل مكان حيث يكون الثابت واجباً :

```
const min = -7; max = 7; type T = array min .. max of integer;
```

8- وَجِبَ تعريف كل الثوابت في بداية البرنامج (أو الفدرة)

5.5- فترة ، تعداد بولّي ، سمة ، جدول ، فقرة

(Intervalle, Énumération, Booléen , caractère, Tableau, Article)

(١) - يقيّد النوع فترة النوع الترتيبي لكنه يحافظ على عملياته :

7.. 18, '0' .. '9' Lundi .. Vendredi (الإثنين ... الجمعة)

١ - يعرف النوع تعداد قيمياً مع ترتيبهن عن طريق التعداد . تكون القيم معرفتين الثابت :

الأحد السبت الجمعة الخميس الأربعاء الثلاثاء الإثنين
(lundi, mardi, mercredi, jeudi, vendredi, samedi, dimanche)
(inférieur, égal, supérieur) أكبر يساوي أقل

- العمليات هي تلك الخاصة بالأنواع الترتيبية : `ord, pred, succ` ، علاقات
- 2 - يتوافق النوع المعروف مسبقاً بولي مع `type boolean = (false, true)` . العمليات هي العمليات المنطقية `and` (و) ، `or` (أو) ، `not` (لا) ، وعمليات الأنواع الترتيبية .
- 3 - النوع المعروف مسبقاً سمة ، `char` ، هو التعداد للسماح الممكن إستعمالها . يتم تعريفه في كل حاسوب بطريقة مختلفة ، لكنه يحتوي على الرموز الخاصة للغة ، على الأحرف ضمن تمثيل للأحرف واحد على الأقل ، مرتبين (`'a' < 'b' ..`) لكن ليس بالضرورة متتالين ، وعلى الأرقام ، مرتبين (`'0' < '1' < ...`) ومتتالين (`succ ('0') = '1' ..`)
- 4 - العمليات هي تلك الخاصة بالأنواع الترتيبية : `ord, pred, succ` ، علاقات ، `chr` و `مُعكس` `ord (chr (n)) = n` .
- 5 - النوع جدول هو مجموعة متغيرات كلها من نفس النوع :
مكون `of` [دليل] `array`
حيث أن الدليل هو نوعاً ترتيبياً والمكونات من نوع أي كان :
`array [0..7] of char`
اختصار : `array [boolean , '0' .. '7'] of reel` مكافئ لـ
`array [boolean] of array ['0' .. '7'] of real`
- 6 - في متغير جدول ، يكون الدليل (indice) كناية عن تعبير يُكتب بين معقّفين : `T [i]`
اختصار : `x [false , '2']` مكافئ لـ `x ['2'] [false]`
الكتابة `x [h]` تعني سطرًا ؛ لا يوجد تنويط للأعمدة .
- 7 - النوع فقرة هو مجموعة متغيرات ، كل منها من نوع أي كان :
`record c1; T1; c2, c3 : T2 end`
يستعمل المشتق منقاة للنيل :
`record a : T1; case b: boolean, of true : (i: integer) ; false: (r: real) end`
- 8 - في المتغير حفل فقرة ، يكون إسم الحقل مسبقاً بنقطة : `Y.c1`
- 6.5 - تعيين (Affectation)
0 - تسمح عبارة التعيين بتعيين قيمة تعبير ، لمتغير من نوع متساوق :
`variable : = expression`
مثال : `x := pi + 1.0` `i := i + 1`
- 1 - يمكن تعيين عدد صحيح لعدد حقيقي ، إن هذا هو حال التغير الأوتوماتي الوحيد .

2 - لكي نعين عدد حقيقي لعدد صحيح ، نستعمل القطع (أو البتر) (trunc) أو التكبير (التدوير) (round)

3 - يكون التعبير مؤلفاً من تعبير بسيط ، أو من تعبيرات بسيطة ومؤثر علاقة (\leq)

(in < > > = > < =

$x * y + z / y$ $x - y = z + y * y$

4 - يتكوّن التعبير البسيط من متأثرات ومؤثرات ضمن إطار ترتيب الأسبقّيات التنازلية :

not

* / and div mod ضرب

+ - or جمعي

5 - ضمن الأسبقية المتساوية ، يتم التقييم من اليسار إلى اليمين :

$x * y + y / z = \text{not } q$ تعني

$((x * y) + (y / z)) = (\text{not } q)$

يسمح وضع الأقواس دائماً بفرض ترتيب الحسابات .

6 - + - * mod div عند تطبيقهم على أعداد صحيحة ، يجعلون النتيجة صحيحة ، مثل

الدوال المعرفة مسبقاً succ, pred, ord, sqr, abs .

trunc و round يقومان بالتحويل من الحقيقي إلى الصحيح ؛ odd (x) هي صحّ إذا

كان x مفرداً ؛ chr (x) تحوّل الصحيح إلى سمة .

7 - + - * عند تطبيقهم على أعداد حقيقية ، يجعلون النتيجة حقيقية ، مثل الدوال المعرفة

مسبقاً abs و sqr . / هو دائماً قسمة حقيقية ؛ sin ، cos ، exp ، ln ، sqrt ،

arctan هنّ دوال ذوات نتيجة حقيقية .

8 - يمكن أن يكون المتغير المعيّن من كل نوع ما عدا السجل : بسيط ، دليل (pointeur)

جدول ، فقرة ، مجموعة ، سلسال ،

7.5 - إذا ، الحالة ، لـ ، طالما ، كرّر ، مع (si, cas, pour, tant que, répéter, Avec)

READ, write

اقرأ ، أكتب

0 - الشرط هو تعبير يُعطي تقييمه إما القيمة صح (true) ، إما القيمة خطأ (false)

1 - في العبارة إذا : if condition then énoncé else énoncé

أو : if condition then énoncé

لا تُنفذ العبارة then إلا إذا كان الشرط صحّاً ، بينما لا تُنفذ العبارة else إلا إذا كان

الشرط خطأ . يمكن أن تكون العبارات بسيطة ، (إذا ، تعيين ، لـ ، ...) أو مركبة (**begin énoncé; énoncé ... end**)
مثال :

if min > T [a] then min := T [a]

2 - في العبارة الجالة

case expression of c1: E1; c2: E2; ...; cn: En end

التعبير ، من نوع ترتيبي ، عليه أن يأخذ قيمة إحدى الثوابت c_i ، تُنفذ عندها العبارة E_i وهي وحدها .
مثال :

**Case comparaison (x, T [c]) of inférieur: b := c ; égal : y := c ;
supérieur : a := c end**

3 - العبارة لـ

for variable := expression 1 expression 2 énoncé

تسمح بتكرار العبارة ، لكل قيم المتغير الذي هو من نوع ترتيبي : عبارة 1 ، succ (عبارة 1) ، succ (... عبارة 2) . إذا كان التعبير $2 >$ التعبير 1 ، فإن العبارة لا تُنفذ . إن إستبدال **to** بـ **downto** ، يحمل على تطبيق **pred** بدلاً من **succ** . بعد التنفيذ ، يكون للمتغير قيمة غير محددة .
مثال :

S := 0 ; for a := 0 to N- 1 do S := S + Z [a]

4 - في العبارة طالما

while condition do énoncé

طالما الشرط صحيحاً ، فإن العبارة تُنفذ .
مثال :

f := 1 ; i := 0 ; while f < 100 do begin i := i + 1 ; f := f * i end

5 - في العبارة كرر

repeat énoncés until condition

تُنفذ قائمة العبارات حتى يصبح الشرط صحيحاً .
مثال :

repeat read (c); x := succ (x) until c = ' '

6 - تُجمّع العبارة مع ، منافذ (نيل) ضمن الفقرة .
مثال :

with $p \uparrow [i]$ **do** $a := b - t$

يتوافق مع $p \uparrow [i].a := p \uparrow [i].b - p \uparrow [i].t$

7 - $read(a, b)$ تقرأ قيمتين على سجل الدّخل **input** وتعيّنهما للمتغيرات a و b يمكن قراءة أعداد صحيحة ، حقيقية أو سمات .

8 - $write(e)$ تكتب قيمة التعبير e على سجل الخرج **output** . يمكن كتابة اعداد صحيحة ، حقيقية ، سمات ، سلاسل ، بولي لكن ليس جدولاً أو فقرة أو مجموعة أو دليلاً (pointeur) . يمكن تحديد حقل الطباعة :

write (car: 3, entier : N, réel : total : dec)

writeln تنهي السطر الجاري العمل فيه على **output** .

8.5 - إجراء ، دالة ، مجموعة ، سلسال ، سجل ، دليل

(Procédure, Fonction, Ensemble, Chaîne, Fichier, Pointeur)

0 - يُعطي تصريح الإجراء اسماً لعبارة مركّبة. يمكن ضمن إجراء، التصريح عن متغيرات، أنواع ، ... (هذه فِدرَة) وإعطاء اسماً لأدوات لا تكون فعلياً معروفة إلاّ عند النداء : وسائط صورية .

مثال :

```
procedure  $P(i, j : integer);$  var  $l, c : integer;$   
begin for  $l := 1$  to  $i$  do begin  
    for  $c := 1$  to  $j$  do write ('★'); writeln end  
end;
```

1 - إن عبارة نداء الإجراء تُنشط الإجراء وتحدّد الوسائط الفعلية ؛ مثلاً : $P(15, 10)$ يمكن أن يكون الوسيط قيمة (قيمة الوسيط الفعلي أُعطيَتْ إلى الوسيط الصُّوري) ، متغيراً (الوسيط الصُّوري يعطي منفذاً إلى الوسيط الفعلي) ، إجراءً ، دالة أو جدولاً ضريباً .

مثال :

```
procedure  $\sigma(a, b : real; var c : real);$  begin  $c := a + b$  end;  
alors  $\sigma(y \star z/2.0, \text{sqr}(i2), x)$  فاذن لـ  
a pour effet  $x := y \star z/2.0 + \text{sqr}(i2)$  المفعول
```

2 - تتصرّف الدالة كالإجراء ، لكن تعطي نتيجة :

مثال :

```
function S (x, y : real) : real begin S := x + y end;
x := S (y ★ z/2.0, sqr (i2))
```

3 - النوع مجموعة هو مجموعة قيم من نوع ترتيبي . تُطَبَّق على المجموعات عمليات
الإتحاد (+) ، التقاطع (*) ، الفرق (-) ، المساواة (= و < >) ، التضمن (= و < =) والإنتهاء (in) .

يقوم مُنشئ المجموعات [...] بتمرير قيم ترتيبيّة إلى المجموعة ؛ تنوُّط
المجموعة فراغ بـ [] .

مثال :

```
var possible set of (noir, jaune, rouge, vert); C : set of char;
if noir in possible then ...
C := ['0'..'9', '.', 'E']; if lu in C then ...
```

4 - سلسال من n سمة هو من النوع

Packed array [1.. n] of char

يمكن كتابته ، تعيينه ، مقارنته .

مثال :

```
software := 'logiciel' avec var software: packed array [ 1..8 ] of char
```

5 - النوع سجل هو تتالي مركّبات ، كلها من نفس النوع ، والتي تمرُّر عليها نافذة بشكل
متتالي ؛ فقط المركب المكشوف من قبل النافذة يمكن بلوغه .

تصريح :

f : file of type du composant

النافذة هي متغيّر ننوِّطه $f \uparrow$.

العمليات :

reset (f) من جديد خُذ موضع البدء

get (f) اقرأ

rewrite (f) أُنلِف ، لكي تكتب

Put (f) اكتب

eof (f) اختبر نهاية السجل

مثال : نسخ السجلات

```
reset (f); rewrite (g); while not eof (f) do begin
```

```
get (f) ; g ↑ := f ↑ ; put (g)
```

```
end
```

6 - النوع المعروف مسبقاً text هو سجل سمات مركب على هيئة سطور . تُركّز نهاية السطر (يحدّد موقعها) ، كتابةً بـ writeln (l) ؛ بينما تُختبر ، قراءةً ، بـ coln (l) ويمكن كذلك قراءة أو كتابة أدوات من نوع غير السمات (سيتم تحويلهم إلى سلسال سمات) : صحيح ، حقيقي . . .

7 - يقدّم الدليل (pointeur) منفذاً إلى متغير تحريكي من نوع محدد .
مثال :

```
type ptr = ↑item; item = record val : real; lien : ptr end
var p, tête : ptr;
p := tête; while p <> nil do begin write (p↑.val); p := p↑.lien end
```

8 - يُخلق المتغير التحريكي بواسطة new ويُتلف بواسطة dispose ؛ لا تخضع مدة حياته إلى قواعد الفدرات .
مثال :

```
new (p); p↑.val := x; p↑.lien := tête; tête := p;
```

الفصل السادس

ملحقات

ملحق 0 : دليل البرامج

- فوترة على ميزان مسجل (3.1) (2.1) (3.1) (3.1)
- جدول المربعات (3.1)
- وسط حسابي لـ n قيمة (3.1)
- حساب الدفع لعامل بالساعة (4.1)
- معادلة من الدرجة الثانية (4.1)
- عدّ القيم الموجبة أو السالبة (4.1)
- جمع قيم ، متبوعة بـ 1 - (5.1)
- الجذر التربيعي على طريقة نيوتن (5.1)
- عدّ التبايدات في نص (5.1)
- فوترة مع تعرفه (6.1)
- وسط حسابي وانحراف معياري (6.1)
- تردد أرقام في نص (6.1)
- معدل علامات (6.1) (6.1)
- حاصل ضرب مصفوفات (6.1)
- م حساب صغير + - * / (6.1)
- حجم برميل (1.2)
- قراءة سجل نص (1.4.2) (2.4.2)
- نسخ سجل (4.4.2) (7.4) (7.4)
- رسم لمنحنى (6.4.2)
- متسلسلة Fibonacci (1.1.3)
- جدول الخطوط الخاص بحساب المثلثات (6.1.3) (2.3.3)

- جمع ساعات (1.2.3)
- العمل الذي يجب القيام به كل يوم (2.2.3)
- تحويل رقمي - عشري (1.3.3)
- تحويل ثنائي - عشري (1.3.3)
- فرز بمبادلات متتالية (3.3.3)
- تردد أحرف في نص (4.3.3)
- كشف مصرفي (1.4.3)
- مدى التصريجات (3.4.3)
- وسائط (4.4.3)
- إجراءاً يحسب مجموع عددين (4.4.3)
- إجراءاً يحسب مجموع متجهين (4.4.3)
- دالة تحسب مجموع عددين (5.3)
- دالة التكامل (1.1.4)
- برنامج للتكامل (1.1.4)
- إجراء مع جدول ضبط (2.1.4)
- القاسم الأكبر المشترك (2.4)
- دالة أكرمان (Ackermann) (2.4)
- البحث عن قيمة الإنتاج القومي الختام لبلد (2.3.4)
- إجراء قراءة سلسال (جدول ضييط) (2.3.4)
- تنقيب فرقاني (2, 3, 4)
- كتابة مجموعة (4.4)
- قراءة مجموعة (4.4)
- رأبخوا دوري فرنسا (6.4)
- فرز ثنائي (6.4)
- تمدد سجل (7.4)
- دمج سجلات مفروزة (7.4)
- إعادة كتابة نص على n عامود (8.4)
- اعداد شبه صدفية (10.4) (10.4)
- التكامل على طريقة مونت - كارلو (10.4)

ملحق 1 - مظاهر داخلية

لا يمكن لحاسوب ، إلا تنفيذ التعليمات المكتوبة في لغة خاصة به : لغة الآلة (تُسمّى في بعض الأحيان بكلمة في غير محلّها : مؤوّل) . عبارة الباسكال ليست مباشرة قابلة للتنفيذ ، يجب في البدء ترجمتها إلى تعليمات في لغة الآلة ، إمّا مباشرة بواسطة مصرّف ، إمّا بطريقة غير مباشرة بواسطة مُفسّر (p - كود ، UCSD . . .) ، أقلّ بظاً من المصرّف .

تبقى التعليمات والمعطيات في الذاكرة ، وحدة النيل في الذاكرة هي الكلمة ، المؤلفة من عدد ثابت من البتات ، كل بته ، أو موقع ثنائي يساوي إمّا 1 إمّا 0 (صفر) ؛ يمكن أن يوجد من 4 إلى 128 بته في الكلمة (عادة 8 ، 16 أو 32 على الميكروحاسوب ، 16 أو 32 على الميني حاسوب ، 32 ، 48 ، 60 أو 64 على الحاسوبات الكبيرة) . تُشغل البايته 8 بته . يمكن نيل كل كلمة فقط بواسطة عنوانها .

يمكن أن تتوافق أنواع المعطيات في الباسكال مثلاً (النظم لا تفرض تمثيلاً موحداً) :

صحيح : على كلمة مكوّدة في القاعدة 2 (مع إصطلاح خاص للأعداد الصحيحة السالبة) . على كلمة من n بته ، يمكن أن تأخذ الأعداد الصحيحة 2^n قيمة ، من $2^{n-1} - 1$ إلى $2^{n-1} - 1$ ، هذا ما يحدّد قيمة \maxint .
حقيقي : على كلمة ، اثنتين أو أربع كلمات تبعاً للحاسوب ، تُمثل كلُّ على حدة الأسّ والجزء العشري : مثلاً بـ 32 بته ، يأخذ الأسّ قيمة تصل إلى 38 والجزء العشري من 6 إلى 7 أرقام عشرية .

بولي : على بته (في كلمة ، تكون البتات الأخرى غير مستعملة)
جدول : على $n \times p$ كلمة في حال وجود n مركّب كل من p كلمة ؛ يتم النيل من مركّب بالتقسيم (indexation) : القيمة المحسوبة للإزاحة من الكلمات في الجدول (دليل) تُضاف إلى عنوان أوّل مركّب .

فقرة : مثل حال الجدول ، لكن يكون الدليل قيمة ثابتة (متوافقة مع إسم الحقل المثال) ، وليس محسوبة .

رصّ : `of boolean [1.10] array` تُشغل 10 كلمات ، لكن تُستعمل فقط بته واحدة بالكلمة ؛ تُشغل `Packed array [1.10] of boolean` عشر بتات متتالية .

مجموعة : متسلسلة بتات متتالية ؛ العمليات على المجموعة (+ - * in) تكون إذن عمليات على البتات . إذا سمح به حجم الكلمة ، فإن حجم المجموعة محدد

(مثلاً مع كلمات من 60 بته ، مجموعات مركبة على 59...0) هذا ما يمنع النوع

set of char ...

سمه : غالباً على بايته (إصطلاح : نشطب الأصفار) .

لعب سمات Display Code Pascal (CDC) : 6 بتات

u \ d	0	1	2	3	4	5	6	7	8	9
0		A	B	C	D	E	F	G	H	I
1	J	K	L	M	N	O	P	Q	R	S
2	T	U	V	W	X	Y	Z	0	1	2
3	3	4	5	6	7	8	9	+	-	☆
4	/	()	\$	=	^	,	.	'	[
5]	:	≠	{	v	^	↑	}	<	>
6	≤	≥	⌊	;						

عدد ترتيبى $u + 10 * d$

مثال : $\text{ord}('A') = 1$ ، $\text{chr}(45) = \text{' '}$ (تباعد)

الأعداد الترتيبية الأكبر من 63 والصفر ليست مُنتقة ، الأحرف متتالية ، لا يوجد

أحرف صغيرة .

لعب سمات ASCII بدون شفعية (Parité) : 7 بتات

x \ y	0	1	2	3	4	5	6	7
0	nul	dle		0	@	P	'	p
1	soh	dcl	!	1	A	Q	a	q
2	stx	dc2	"	2	B	R	b	r
3	ext	dc3	#	3	C	S	c	s
4	eot	dc4	\$	4	D	T	d	t
5	enq	nak	%	5	E	U	e	u
6	ack	syn	&	6	F	V	f	v
7	bel	etb	'	7	G	W	g	w
8	bs	can	(8	H	X	h	x
9	ht	em)	9	I	Y	i	y
10	lf	sub	☆	:	J	Z	j	z
11	vt	esc	+	;	K	[k	{
12	ff	fs	,	<	L	\	l	
13	cr	gs	-	=	M]	m	}
14	so	rs	.	>	N	↑	n	~
15	si	us	/	?	O	—	o	del

عدد ترتيبي = 16 * x + y

مثال 65 = ord('A') ، chr(32) = ' ' (تباعداً)

الأحرف متتالية ؛ نُمَيِّز بين الأحرف الصغيرة والكبيرة . تكون سمات العدد الترتيبي من 0 إلى 31 ، و 127 غير قابلة للطباعة ، إنها سمات مُحْكَم مُستعملة لإرسال المعطيات (cr = عودة (return) ، bel = إنذار (alarm) ، bs = عودة إلى الوراء (...., retour arriere) .

لعب سمات EBCDIC 8 0 بته

ضمن السمات القابلة للطبع ، نجد

		espace	64	a	129	n	149	A	193	N	213
0	240	.	75	b	130	o	150	B	194	O	214
1	241	(77	c	131	p	151	C	195	P	215
2	242	+	78	d	132	q	152	D	196	Q	216
3	243	☆	92	e	133	r	153	E	197	R	217
4	244)	93	f	134	s	162	F	198	S	228
5	245	;	94	g	135	t	163	G	199	T	227
6	246	/	97	h	136	u	164	H	200	U	228
7	247	,	107	i	137	v	165	I	201	V	229
8	248	-	110	j	145	w	166	J	209	W	230
9	249	:	122	k	146	x	167	K	210	X	231
		,	125	l	147	y	168	L	211	Y	232
		=	126	m	148	z	169	M	212	Z	233

succ('i') ≠ 'j'

succ('r') ≠ 's'

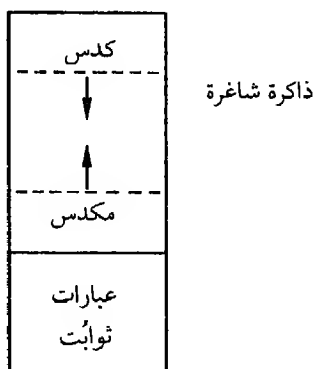
(espace · تباعد)

مكدس وكُدُس (pile et tas)

عند بداية التنفيذ ، تُوجد فقط المتغيرات المصرحة في مستوى البرنامج ، إذن الإجمالية لكل الإجراءات والدوال ، وكذلك السجلات input و output . عند تنشيط إجراء ، أو دالة تُخلق متغيراتها الموضعية ؛ يتم فيما بعد إتلافهن عند نهاية التنشيط (عند « العودة إلى المنادي ») : تكون فدرات الذاكرة مُكدسة ، ومن ثم مزالة ، ضمن منطقة في الذاكرة تُسمى مكدس . في كل لحظة ، تكون بذلك المتغيرات الموضعية المُمكن بلوغها في القدرة في قمة المكدس ؛ تُحدّد بذلك مدة حياتهم بمدة تنشيط الإجراء ، أو الدالة ، هذه التقنية بتكرار النداءات .

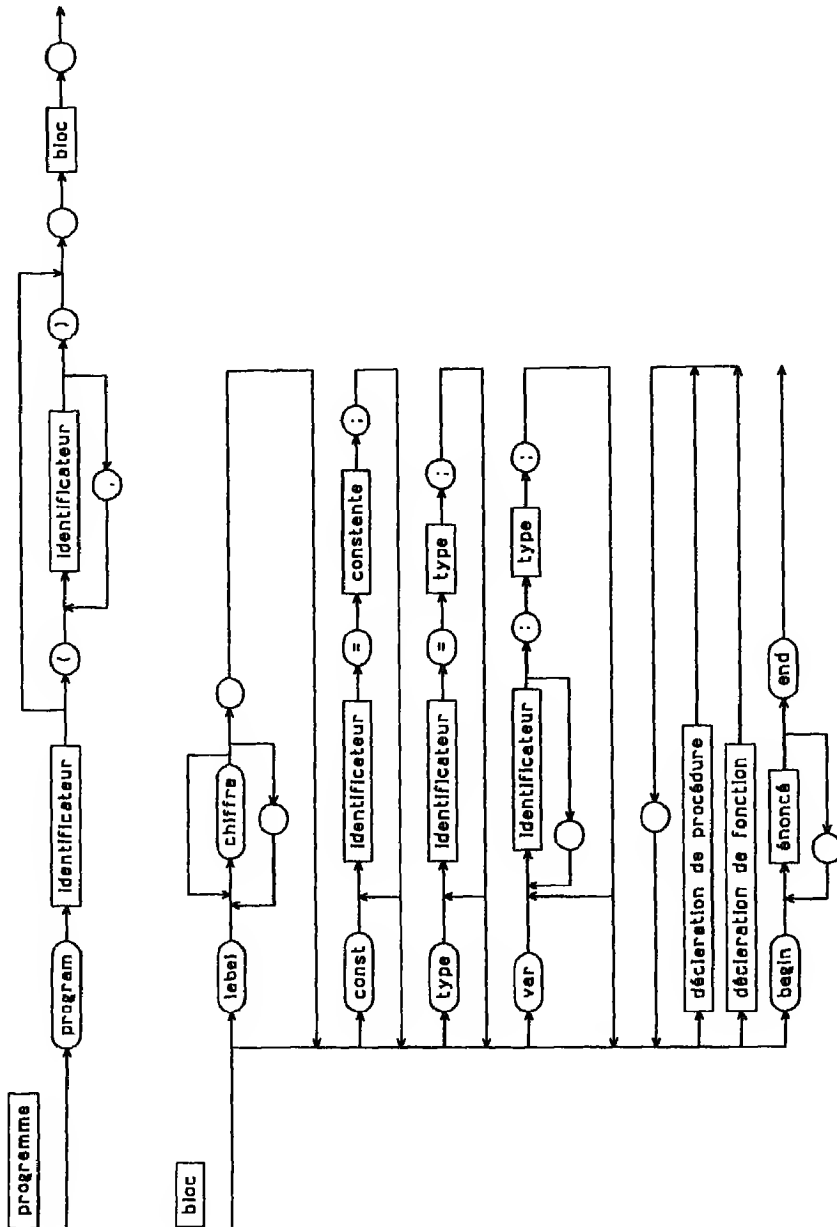
ملاحظة : يُعرّف حجم كل فدرية من المتغيرات الموضعية قبل تنفيذ البرنامج ، هذا

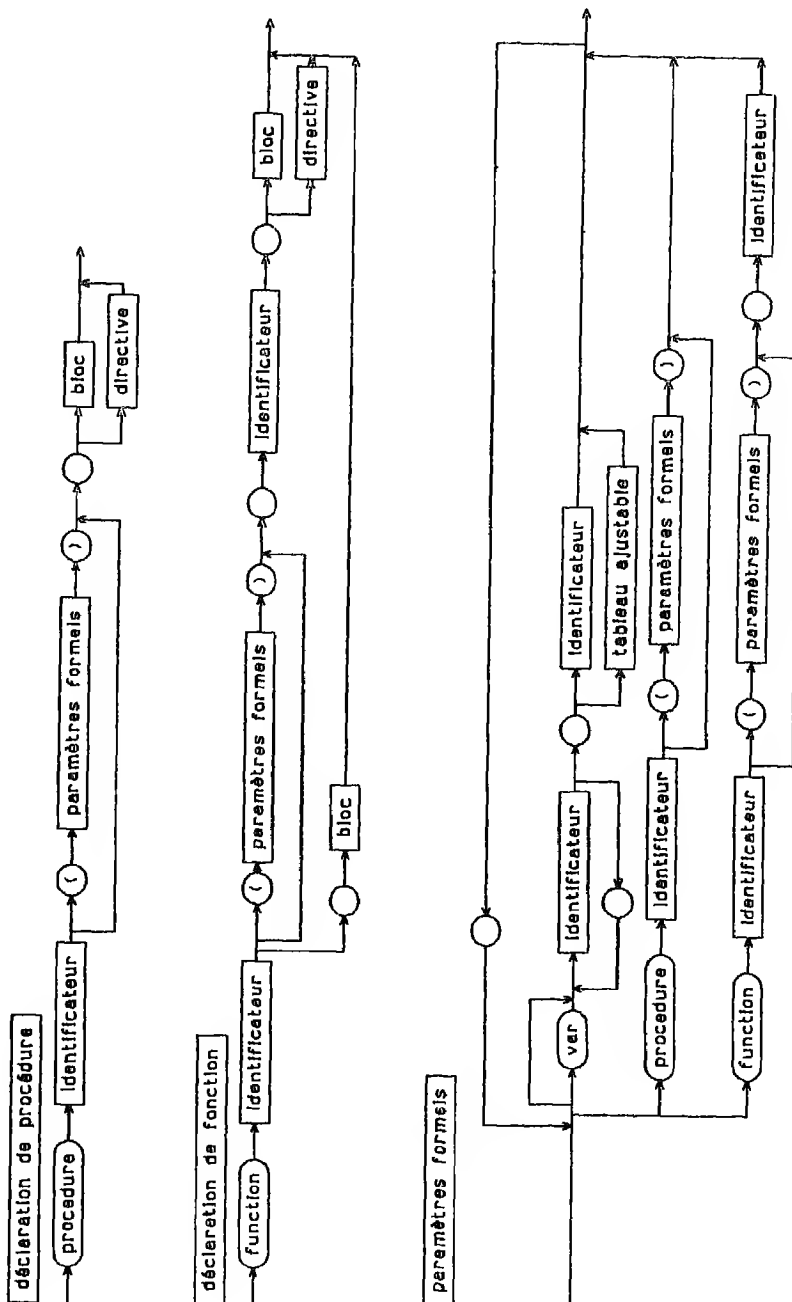
ما يسمح بإدارة النداءات الفعالة خاصة وهذا ما يستتبع تنفيذ سريع .
 في المقابل ، يكون للمتغيرات التحريكية (المخلوقة بـ new) مدة حياة تمتد من لحظة خلقهن (new) حتى نهاية تنفيذ البرنامج ، أو حتى إتلافهن المتعمد (بواسطة dispose) : لا يمكنهن البقاء في المكس . يتم ترتيب هذه المتغيرات التحريكية ضمن منطقة في الذاكرة تسمى كُدس مميّزة عن المكس .
 ليس من علاقة ما بين التطور لكل من المكس والكس . لكي نؤمن إشغال أفضل للذاكرة الشاغرة ، نفضل عدم الإشغال الثابت للكس والمكس (يمكن أن يكون المكس مشبعاً بينما يكون الكس فارغاً !) ، بل بالأحرى العمل على توسيعهما كدس ومكس باتجاه متعاكس :

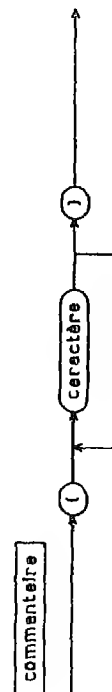
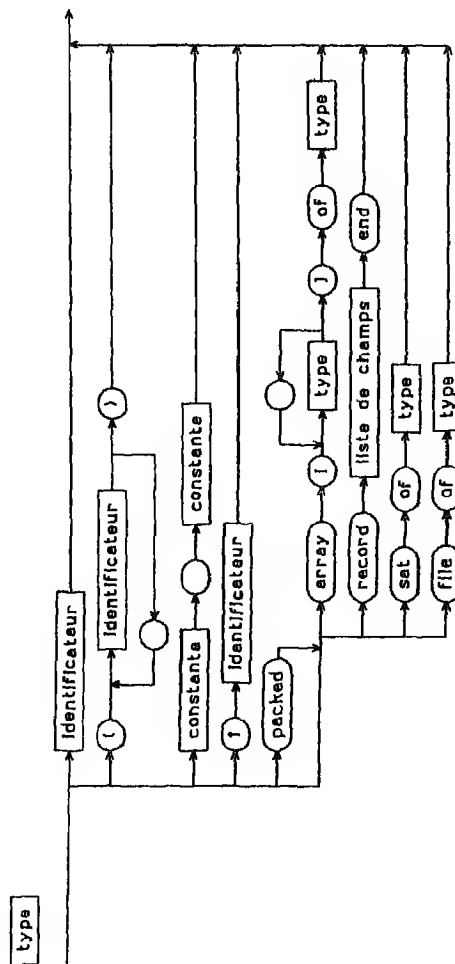
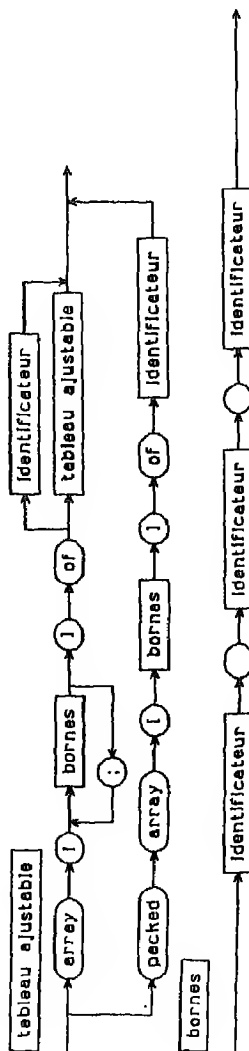


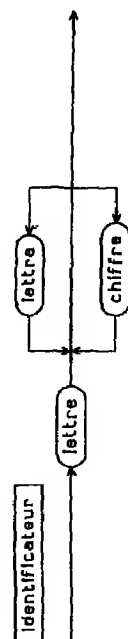
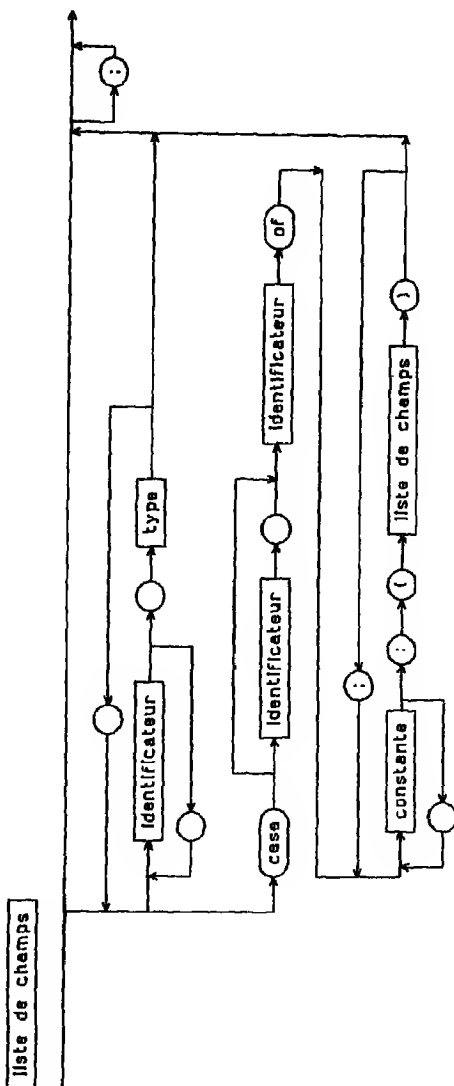
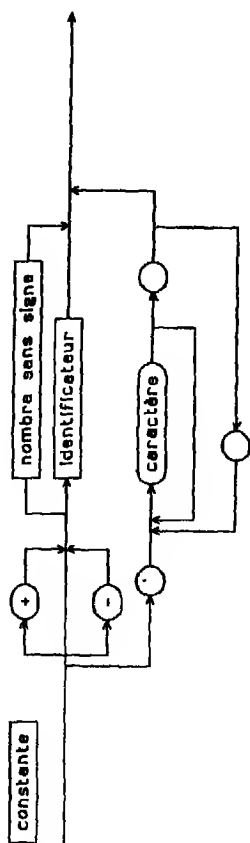
يتوقف التنفيذ إذا ما إلتقى الكس بالمكس لأنه يكون قد غدت حينها كل الذاكرة الشاغرة ، مُشبعة .

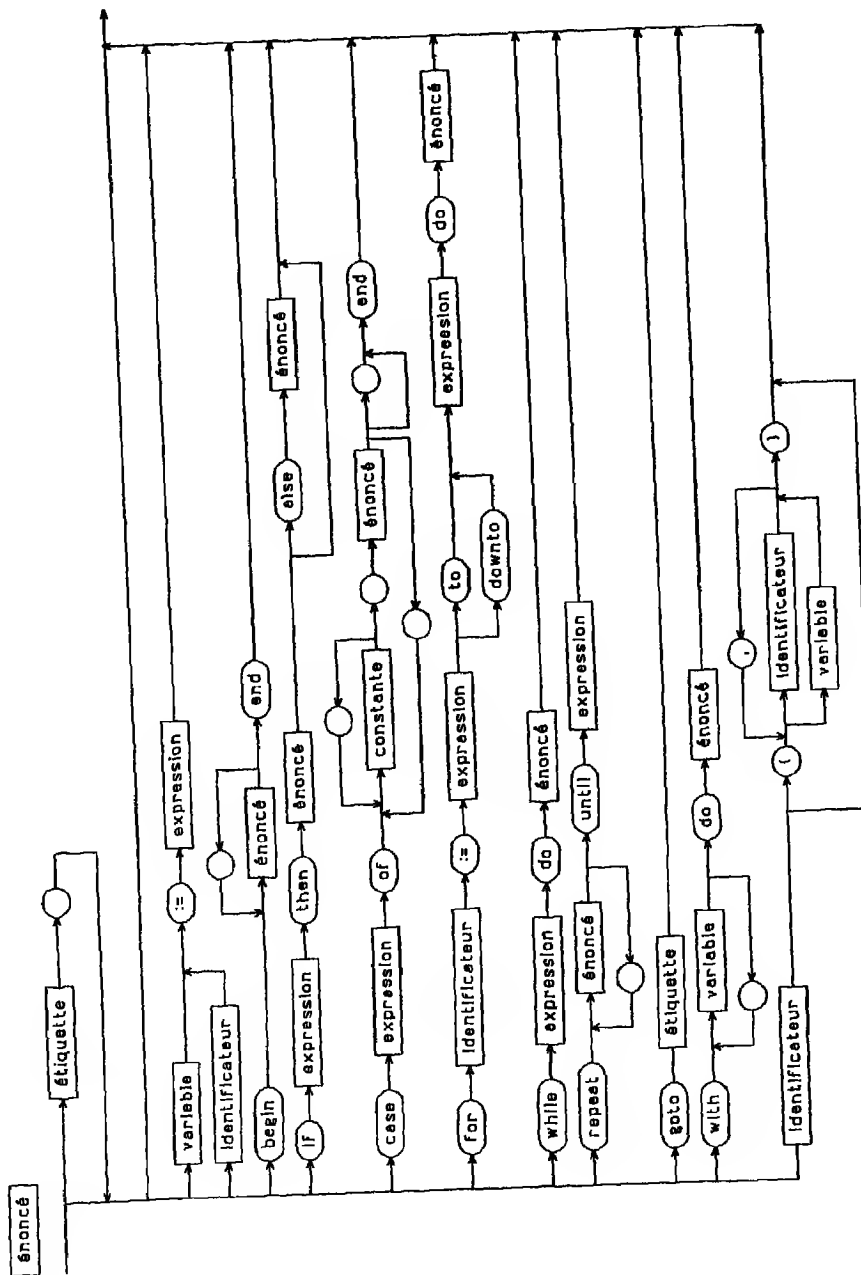
ملحق 2 : مخططات النحو

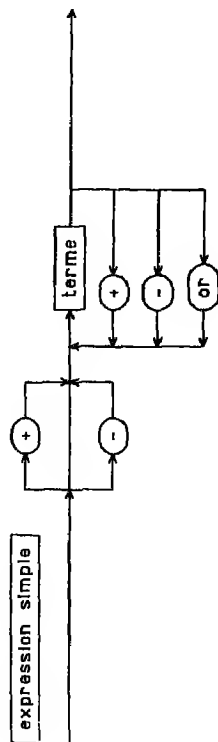
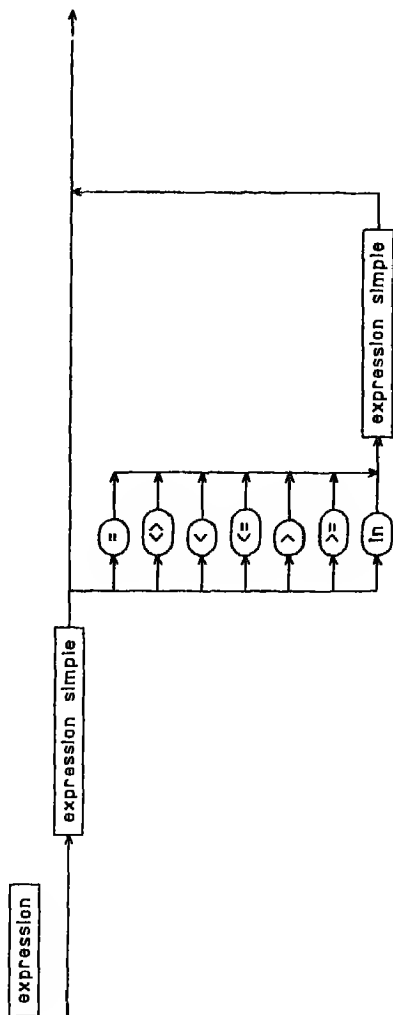
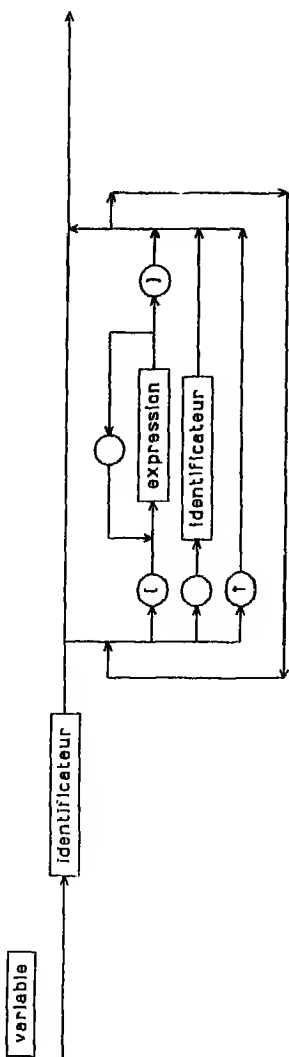


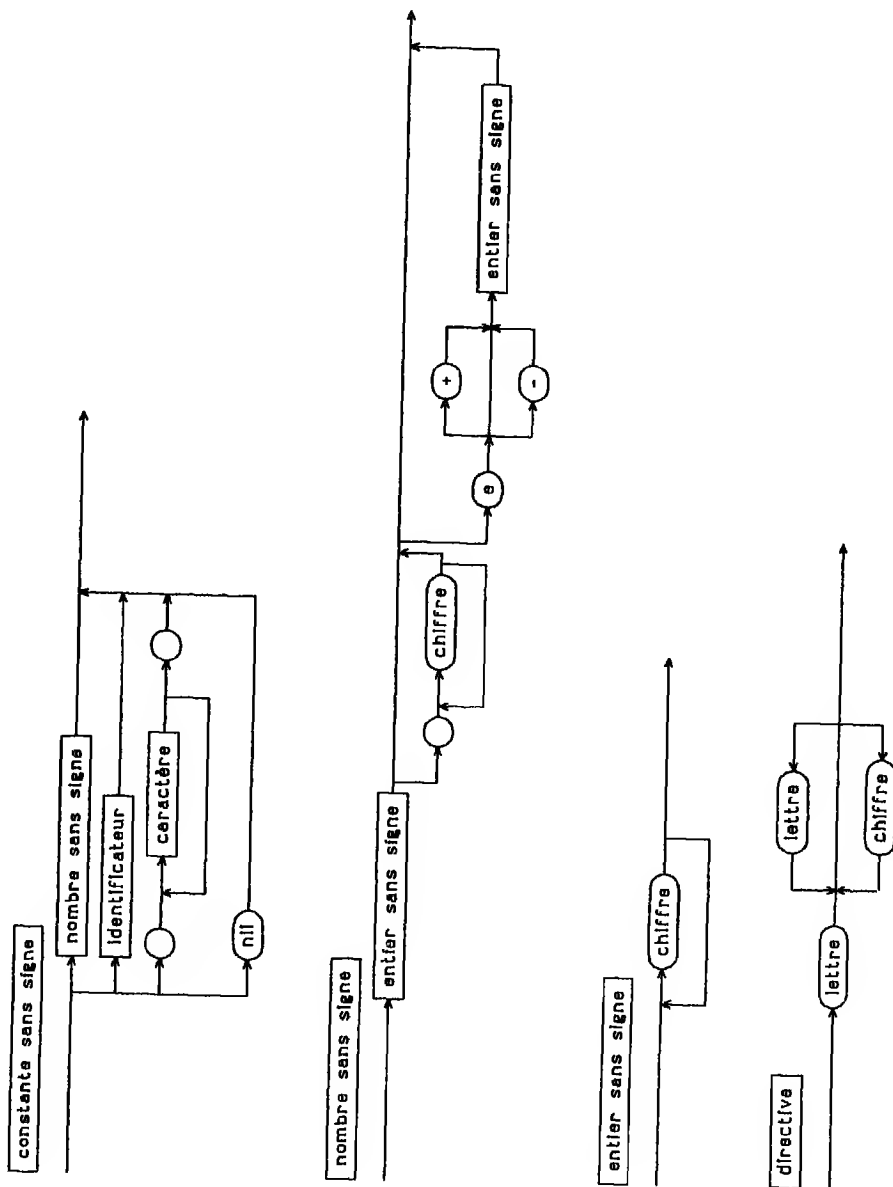












ملحق 3 : تمثيل مادي

رموز خاصة

+ - ★ / = ' [] . > = , < >
; : ↑ () < > <= >= <>
:= espace { }

(espace : تباعد)

تمثيلات إضافية

↑ ^ o @
u
[({ (★
]) } ★)

كلمات دليلية

A	and	end	N	nil	S	set
B	array	F	file	not	T	then
C	begin		for	O		to
	case		function	or		type
	const	G	goto	P	packed	U
D	div	I	if	procedure	V	var
	do		in	program	W	while
	downto	L	label	R		with
E	else	M	mod	repeat		

لا يمكن أن يكون لدى أيّ معرف ، كتابة الكلمات الخاصة بإحدى هذه الـ 35 كلمة دليلية .

معرفين معرفين مسبقاً

يمكن إعادة التصريح عنهم في البرنامج ويكون لهم بذلك مدلول آخر .

نوابت

false خطأ (بولي)

true صح (بولي)

maxint الأعداد الصحيحة محدّدة بـ maxint ... maxint

أنواع

(false, true) =	boolean
تُعب سمات ، يتعلق بالحاسوب (ملحق 1)	char
صحيح	integer
حقيقي	real
سجل نص . مثلاً : input و output	text

متغيرات

سجل موحد حيث يتم قراءة المعطيات	input
سجل موحد حيث يتم كتابة النتائج	output

توجيهات

يفصل عنوان وجسم الإجراء : تكرارية متقاطعة	forward
يحدد بأن جسم الإجراء أو الدالة هو خارجي عن البرنامج : « تعريف منفصل » ليس بالضرورة أن يعرف الـ external في كل حاسوب	external

دوال

(x صحيح أو حقيقي ، نتيجة من نفس النوع)	abs (x)
قوس ظل x	atan (x)
سمة العدد الترتيبي x (ملحق 1)	chr (x)
جيب التمام للزاوية x	cos (x)
بولي ، نهاية السجل	eof (f)
بولي ، نهاية السطر في سجل نص	coln(f)
e ^x	exp(x)
لوغاريتم نيبري لـ x	ln (x)
بولي ، مفردة العدد الصحيح x (صح إذا x مفرداً)	odd (x)
العدد الترتيبي لـ x (من نوع ترتيبي)	ord (x)
سلف x (من نوع ترتيبي)	pred (x)
قيمة صحيحة مدورة للعدد الحقيقي x (العدد الصحيح الأقرب)	round (x)
جيب الزاوية x	sin (x)
x ² (عدد صحيح أو حقيقي ، نتيجة من نفس النوع)	sqr (x)
\sqrt{x}	sqrt (x)
خلف x (من نوع ترتيبي)	succ (x)
القسم الصحيح للعدد الحقيقي x	trunc (x)

إجراءات

تحرّر المتغير المدلّل عليه بـ p	dispose (x)
قدّم النافذة $f \uparrow$ واقرأ	get (f)
خلق متغيّر مدلّل عليه	new (p)
رصّ	pack (...)
تخطّي الصفحة	page (f)
اكتب ، وقدم النافذة $f \uparrow$	put (f)
$V := f \uparrow$; get (f)	read (f, v)
ركّز النافذة في بداية السطر التالي	readln (f)
أعد تركيز السجل عند بدايته ، الشأن معاينة	reset (f)
أتلّف السجل ، انتقل إلى الشأن نتائج	rewrite (f)
تفكيك	unpack (...)
$f \uparrow := c$; put (f)	write (f, c)
اكتب نهاية سطر	writeln (f)

ملحق 4 : مشتقات فرنكوفون (بالفرنسية) مع ترجمتها

كلمات دلالية

A and	et	N nil	و	nil	صفر
array	tableau	not	جدول	non	لا
B Begin	début	O of	بدء	de	من
C case	cas	or	حالة	ou	أو
Const	const	P packed	ثابت	paquet	مرصوص
D div	div	procedure	قائمة صحيحة	procédure	إجراء
do	faire	program	إفعل	programme	برنامج
downto	bas	R record	أسفل	article	فقرة
E else	sinon	repeat	والآ	répéter	كرّر
end	fin	S set	نهاية	ensemble	مجموعة
F file	fichier	T then	سجل	alors	إذن
for	pour	to	إلى	haut	إلى
function	fonction	type	دالة	type	نوع
G goto	allera	U until	إذهب إلى	jusque	حتى
I if	si	V var	إذا	var	متغير
in	dans	W while	في	tantque	طالما
L label	étiquette	whith	وسم	avec	مع
M mod	mod		شان		

معرفين معرفين مسبقاً

					ثوابت
false	faux	خطأ	true	vrai	صح
maxint	entmax	تحديد للأعداد الصحيحة			
					أنواع
boolean	booléen	بولي	real	réel	حقيقي
char	car	سمة	text	texte	نص
integer	entier	صحيح			

متغيرات خروج

input	entrée	دخّل	output	sortie
-------	--------	------	--------	--------

		الأبعد	external	externe	توجيهات خارجي
forward	plus loin				
دوال					
abs	abs	قيمة مطلقة	ord	ord	عدد ترتيبي
arctan	arctan	قوس ظل	pred	pred	سلف
chr	carac	سمة	round	arrondi	مُدَوَّر
cos	cos	جيب التمام	sin	sin	جيب
eol	fdf	نهاية السجل	sqr	carré	مربع
	eoln	نهاية السطر	sqr	rac 2	جذر تربيعي
exp	exp	أُسِّي	succ	succ	خَلَفَ
ln	ln	لوغاريتم نيبري	trunc	trunc	قسم صحيح
odd	impair	مفرد			
إجراءات					
dispose	libérer	حرر	read	lire	إقرأ
get	prendre	خُذْ	readln	lireln	إقرأ سطرأ
new	créer	أُخْلِقْ	reset	relire	أعد القراءة
pack	tasser	كَوِّمْ	rewrite	récrire	أعد الكتابة
page	page	صفحة	un pack	détasser	فكك
put	mettre	ضَع	write	écrire	اكتب
			writeln	écrireln	اكتب سطرأ

ملحق 5 مراجع

يوجد تعريف للغة الباسكال في

- * **Langage de programmation - pascal** (1984) : AFNOR Z 65 - 300 **النظم**
الذي هو ترجمة للنظم ISO رقم 7185 . إنه مرجع كامل ودقيق ، لكنه موجه إلى
المنفذ أكثر منه للمستعمل : التعريف الأول للغة الباسكال أعطي من قبل ن. ويرث :
K. Jensen, N. Wirth; Manuel de l'utilisateur Pascal; Masson (1978) *
(بالفرنسية) (ترجمة لـ (User Manual , Revised Report)
ومتّهم بمقال يعرض تعريفاً صورياً للغة ، فإذن صعب القراءة :
- C.A.R. Hoare, N. Wirth; **An Axiomatic Definition of the Programming Lan-**
guage Pascal; Acta Informatica 2, 335- 355 (1973)
(بالإنكليزية) لغة الباسكال متأية من أفكار مطروحة في :
E.W. Dijkstra, C.A.R. Hoare, O. Dahl; **Structured Programming ***
Academic Press (1972)
(بالإنكليزية) ومتّمة على شكل تطبيقي أكثر بطريقة البرمجة بالدقة المتتالية :
N. Wirth ; **Introduction à la Programmation systématique; Masson (1976) ***
(بالفرنسية) نجد وصفاً كاملاً للطريقة الإستنتاجية في :
A. Durcin; **Programmation : Tome 1, du problème à l'algorithmique Tome 2, de**
l'algorithmique au Programme; Dunod (1984)
(بالفرنسية) كثير من أمثلة البرمجة في الباسكال مشروحة في :
P. G. Grogono; **Programming in Pascal; Addison Welsen (1978) ***
(بالإنكليزية) . عملٌ جيد وسهل حول فن البرمجة الجيدة هو :
H.F. Ledgard; **Proverbes de Programmation; Dunod (1978) ***
(بالفرنسية) . بينما يهتم الكتاب التالي أكثر بالمظهر التقني لمجمّعات المعطيات
والخوارزميات :
- N. Wirth; **Algorithmes + Data Structures = Programs; Prentice Hall (1976) ***
(بالإنكليزية) . هذا الأخير يتناول مسائل التصريف المُعالجة بشكل مفصّل أكثر في :
B. Levrat, D. Thalmann; **Conception et inplantation de langages de Prog-**
rammation : une introduction à la compilation, Gaëtan Morin
(بالفرنسية) . مثال بسيط لمصرّف الباسكال مشروح في المقتاتين :

K.M. Chung, H. Yuen; **A Tiny Pascal Compiler**; BYTE 3, 9 (Sept. 78) - 13, *
10 (octo. 78)

نجد معلومات حول إستعمالات الباسكال وتطوره في المجلة
(شهرية بالإنكليزية) للـ ACM ، والمجلة المخصصة للغة الباسكال :
(تصدر كل ثلاثة أشهر بالإنكليزية) **Pascal News**

* أخيراً على كل مبرمج قراءة :

* (صدرت ثلاثة أجزاء منه بالإنكليزية) ؛ **D. Knuth; The Art of Computer Programming**; Addison Wesley (1968).

ملحق 6 : تمديدات

- يرجع نجاح لغة الباسكال في جزء كبير منه إلى بساطتها (نسبة إلى لغات أخرى) .
- كذلك هذه البساطة هي التي تحمل كل مستعمل على إدخال تمديداته الخاصة ، بشكل فوضوي في بعض الأحيان ينتج عنه لغة لا تحمل من الباسكال أو تكاد إلا الاسم التجاري . إلا أن هناك إجماعاً يستخلص حول بعض التمديدات :
- إضافة جزء else أو otherwise على العبارة الحالة (cas) ، جامعةً بشكل ضمني كل الحالات غير المحددة بثوابت الحالة ؛
- إضافة فترات ثوابت الحالة ، في العبارة الحالة ؛
- تصريح منفصل ؛ إستقدام وتصدير ثوابت وإجراءات ؛
- نوع سجل ذي نيل مباشر (متعلق بنظام التشغيل) ؛
- تدميث متغيرات البرنامج عند التصريح ،
- متغيرات دائمة في الإجراءات ، تحفظ قيمها من تنشيط لآخر ؛
- ثوابت مركبة (جدول ، فقرة ، ...) ؛
- تطبيق الدالة ord على الأدلاء (pointeurs) ؛
- نوع سلسل سمات ذي طول متغير ؛
- دالة تحويل عامّة بين الأنواع ؛
- ثوابت ثمانية أو سادس عشرية
- نوع ألفا (alpha) = (10...1 أو) of char [1..8] Packed array ؛
- بلوغ تعليمات لغة الآلة .
- إلخ ...

في نهاية تشرين أول من العام 1985 ، قامت الـ ISO (المنظمة العالمية لتوحيد القياسات) بالتصويت ، من قبل الدول الأعضاء فيها ، على مشروع دراسة النظم « تمديدات الباسكال » . وذلك بهدف تجنب إستمرار الخلافات الحاضرة ، المراد من ذلك هو التعريف السريع لمستوى جديد للغة ، متساوق مع المستويات الحالية 0 و 1 وحاول على عدد من التمديدات :

● معايير : بهدف التعريف المنفصل لأجزاء البرنامج وعدم جمعها في برنامج كامل إلا في لحظة التنفيذ . سيسمح هذا المفهوم بخلق مكاتب معايير ، تحوي الإجراءات والدوال التي يستطيع كل برنامج إستعمالها دون إعادة كتابتها ، ودون الحاجة إلى معرفة تفاصيلها ، بل الحاجة فقط إلى معرفة عملها الوظيفي وطريقة مناداتها . بالنسبة لمبرمج المعيار الذي ليس بحاجة إلى معرفة تفاصيل البرنامج المستعمل ، فعليه من جهته فقط

معرفة الوسائط المفروض مبادلتها مع هذه البرامج والمعالجة الواجب إتمامها . مثلاً يمكن أن يحتوي معيار معين على الإجراءات والدوال المصرحة حالياً مسبقاً : \sin , \cos , abs . . .

● سلاسل سمات : نوع جديد ، مؤلف من سلاسل سمات بحيث يمكن لحجمها التغير خلال تنفيذ البرنامج الذي سيُطبق عليه العمليات التالية :

- تنضيد (من يُنضد) (concaténation) ، منوطة + ، للّصق طرفاً بطرف ، لسلسالين .

- تقسيم (indexation) : $S[i]$ هو السمة i (ieme) من السلسال S

- دالة $\text{length}(s)$ تحسب الطول المتداول للسلسال S

- دالة $\text{Capacity}(s)$ تحسب الطول الأقصى لسلسالٍ مُتساوق بالنسبة للتعين مع السلسال s

- دالة $\text{Position}(s1, s2, i)$ ، وتنتج القيمة 0 إذا لم يظهر السلسال $s1$ في السلسال $s2$ ، أو تعطي موقع أول ظهور للسلسال $s1$ في السلسال $s2$ إنطلاقاً من $s2[i]$

- إجراء $\text{insert}(s1, s2, i)$ ، يولج السلسال $s1$ في السلسال $s2$ إنطلاقاً من $s2[i]$

- إجراء $\text{delete}(s, i, n)$ يحذف n سمة في السلسال s إنطلاقاً من $s[i]$

- إجراء $\text{extract}(s1, s2, i, n)$ يكوّن السلسال $s2$ من n سمة مأخوذة في السلسال $s1$ إنطلاقاً من $s2[i]$

● سجلات ذوات نيل مباشر : النوع الجديد

$\text{file}[n] \text{ of } ..$

ستتوافق مع سجل من n مركّب ، بحيث يمكن بلوغ كل مركّب مباشرة (بواسطة get أو put) .

● سجلات خارجية : حالياً وحدها وسائط البرنامج تسمح بإقامة علاقة بين سجلات البرنامج والسجلات المعروفة لنظام التشغيل . المقصود هو إمكانية إقامة هذه العلاقة خلال تنفيذ البرنامج ، عن طريق إعطاء الاسم نظام السجل كوسيط لإجراء الفتح reset أو rewrite .

● مخططات : ستكون أنواع جداول جديدة بحيث يتم تعريف فترة الدليل خلال التنفيذ ، هذا الحل القاضي بتوسيط (من وسيط) الأنواع هو أكثر شمولية من الجداول الضبيطة .

هذه التمددات ، في حال إتمام تنظيمها ، تحمل إمكانيات جديدة إلى لغة الباسكال ، مع المحافظة على صحّة البرامج الموجودة سابقاً : إنها لا تعرّف لغة جديدة .

غير أنّ عدة لغات تحمّل من قبل تمديدات إلى الباسكال ، لكن دون التأكد من

التواصل مع اللغة الحالية . هكذا فإن البرمجة في لغة الـ Ada أو الـ Modula 2 تسمح بتناول حقولاً سيكون الباسكال فيها غير فعال ، مع المحافظة على نحو قريب من الباسكال .

فهرست

الموضوع	الصفحة
مقدمة	5
الفصل الأول : كيفية البدء بكتابة البرنامج	7
0.1 - التحليل	7
1.1 - الانطلاق من النتيجة	7
2.1 - البرمجة في لغة الباسكال	7
3.1 - التكرارية مع عداد	14
4.1 - شرطي	24
5.1 - تكرارية مع توقف	28
6.1 - جداول	35
7.1 - اختيار طريقة	47
8.1 - تمارين	47
الفصل الثاني : قواعد اللغة	49
0.2 - كتابة برنامج	49
1.2 - مفاهيم مبدئية	50
2.2 - التكوين الإجمالي للبرنامج	53
3.2 - الأدوات المعالجة	56
0.3.2 - أنواع	56
1.3.2 - ثوابت	56
2.3.2 - متغيرات	59
3.3.2 - أنواع بسيطة	60
4.3.2 - تعريف نوع	63
5.3.2 - تكوين الجدول	64

66	6.3.2 - تكوين الفقرة
68	7.3.2 - قواعد التساوق
69	4.2 - الدخل - الحراج
77	الفصل الثالث : قواعد اللغة - معالجات الأدوات
77	0.3 - عبارات
80	1.3 - تعيين - تعبير
89	2.3 - الشرطيات
93	3.3 - طريقة تكرارية
102	4.3 - إجراء
110	5.3 - دوال
112	6.3 - تمارين
117	الفصل الرابع : مفاهيم أكثر تقدماً
117	0.4 - مقدمة
117	1.4 - وسائط
123	2.4 - التكرارية
126	3.4 - الرص - سلاسل السمات
133	4.4 - مجموعات
136	5.4 - فقرات مع مشتقات - عبارة مع
139	6.4 - ادلاء ومتغيرات تحريكية
146	7.4 - سجلات
150	8.4 - سجلات النص
154	9.4 - التسلسل العشيري للأنواع
155	10.4 - متممات
159	11.4 - تمارين
161	الفصل الخامس : مذكرة مساعدة
161	0.5 - برنامج - فدر - مدى - منطقة
162	1.5 - معرف - رمز - فاصل
163	3.5 - أنواع
164	4.5 - ثوابت - أعداد حقيقية - أعداد صحيحة - سلاسل

164	5.5 - فترة - تعداد بولي - سمة - جدول - فقرة
165	6.5 - تعيين
166	7.5 - إذا - الحالة - ل - طالما - كرر - مع - اقرأ - اكتب
168	8.5 - إجراء - دالة - مجموعة - سلسال - سجل - دليل

171	الفصل السادس : ملحقات
171	ملحق 0 - دليل البرامج
173	ملحق 1 - مظاهر داخلية
177	ملحق 2 - مخططات النحو
185	ملحق 3 - تمثيل مادي
188	ملحق 4 - مشتقات فرنكوفون (بالفرنسية) مع ترجمتها
190	ملحق 5 - مراجع
192	ملحق 6 - تمديدات

هذا الكتاب

ليس هناك من شك في أن لغة الباسكال أصبحت إحدى أهم لغات البرمجة في عصرنا الحاضر .

إن المهم الأساسي للمبرمج هو إتباع لغة برمجة معينة تقيه أخطار الوقوع المتتالي في الخطأ عن طريق نسيان تعريف معرف ما أو استعمال إجراء من برنامج ما في برنامج الخاص دون الأخذ بالاعتبار لحيثيات كثيرة .

لعل أهم حسنات لغة الباسكال تكمن في أنها تفرض كلغة على المبرمج إتباع طرق معينة تجعله بمنأى عن الوقوع في الأخطاء أو تساعد على إكتشافها سريعاً في حال وقوعها . إنها لغة تجبر المبرمج على إتباع طريقة البرمجة المركبة التي تسهل كثيراً إمكانية القراءة السريعة للبرنامج وتساعد أكثر على إكتشاف الخطأ وتهوّن على المبرمج عملية استعمال إجراءات موجودة في برامج أخرى .

هنا تكمن أهمية هذا الكتاب الذي يعرض بصورة شاملة ومفصلة للغة الباسكال مع كل تعليماتها وأدواتها ، ويستجيب لطلبات المتهنين الذين يودون تعميق معلوماتهم . كما أنه يمكن الاستفادة منه كمستند تعليمي قائم بحد ذاته للغة الباسكال .